THE ABSOLUTE-TIME CLOCK AND HOW TO BUILD IT

WHAT COMPUTERS STILL CAN'T DO

THE SUPER OTHELLO PROGRAM

"DADDY, IS IT THE PET?"

ROM

COMPUTER APPLICATIONS FOR LIVING

CLOSE ENCOUNTERS

The Mothership

Centerfold

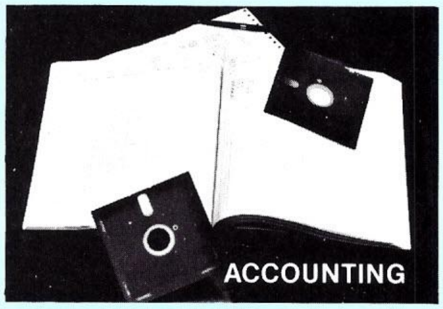
Volume I, Number 9
March-April/$2.00

SWTP CT-64
CHAIN

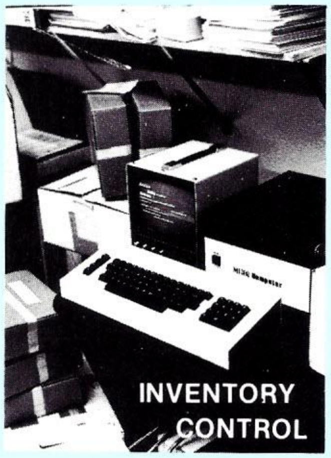
INTERFACE AGE™ MAGAZINE PRESENTS

MICRO BUSINESS '78™

CONSUMER SHOW


ACCOUNTING

EXECUTIVE DATA ACCESS


INVENTORY CONTROL

DATE: **MARCH 17, 18, 19, 1978**

PLACE: **PASADENA CONFERENCE CENTER**

PASADENA, CALIFORNIA

MICRO BUSINESS '78™ will provide a series of marketing forums and exhibits to introduce the small independent businessman to the new low-cost, high-power business microcomputer that will reduce his company's costs, place him in a more flexible marketplace and provide timely data information.

Emphasis will be on the small budget requirements for purchase of an in-house computer. The show will demonstrate the latest systems, exhibiting complete hardware and software from small hand-held programmable calculators to full turn-key computers.

- Latest in Word Processors
- Newly-Released Business Software
- Low-Cost Text Editing Typewriters
- Modularized Computers

THE LOW COST, dependability, simplicity of operation, and cost savings advantages of microcomputers will be discussed in a series of lectures to remove the many misconceptions the average businessman may have about the microcomputer technology. Lectures by such companies as IBM, Commodore Business Machines and Radio Shack will present the businessman with the latest information about application, service and investment.

Author Adam Osborne will discuss business software.

OTHER LECTURES on the program include:

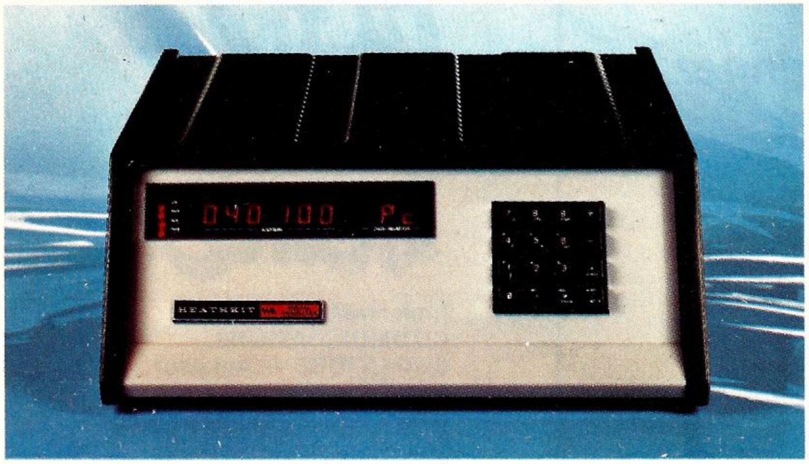
- Small Business Computing Systems
- Evaluating Your Business Computer Needs
- Software Companies
- The Mainframe Companies & The Small Computer
- The Small Business Computer Company
- Computer Stores and the Small Business System
- Retail Mass Marketing of Microcomputers

Sponsored by: INTERFACE AGE Magazine

EXHIBITORS: PLACE YOUR RESERVATION NOW!

Produced & Managed by:
Show Company International
8687 Melrose Avenue
Los Angeles, California 90069
(213) 659-2050
Ed Tavetian

This 8-bit machine, by itself, is as versatile as a lot of systems that include peripherals



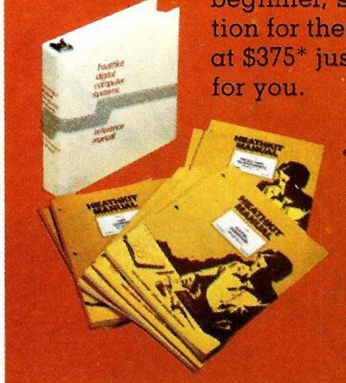
Skeptical? For starters, because of its unique design, the H8 is the only machine in its price class that offers full system integration, yet, with just 4K of memory and using only its "intelligent" front panel for I/O, may be operated completely without peripherals!

Memory Display

Register Display

I/O Port Display

In addition, by using the features of its built-in PAM-8 ROM panel control program, the H8 actually allows you to dig in and examine machine level circuitry. Responding to simple instructions, the "intelligent" panel displays memory and register contents and lets you inspect and alter them even during operation. And for greater understanding, the front panel permits you to execute programs a single instruction at a time. The H8's memory is fully expandable, its 8080A CPU extremely versatile, and with the addition of high speed serial and parallel interfacing you gain the added flexibility of I/O operation with tape, CRT consoles, paper tape reader/punches, and soon floppy disk systems! The H8 offers superior documentation including complete step-by-step assembly and operation manuals, and comes complete with BASIC, assembler, editor, and debug software that others charge over $60 for! H8, simplicity for the beginner, sophistication for the expert and at $375* just right for you.

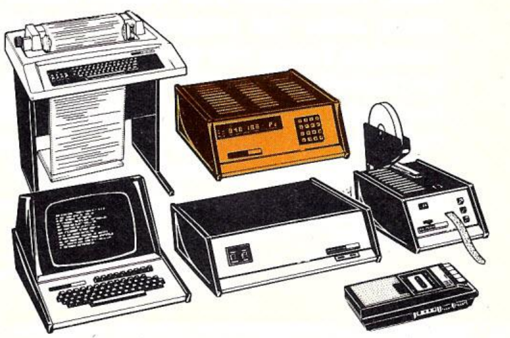


*Prices are mail order net FOB, Benton Harbor, Michigan. Prices and specifications subject to change without notice.

HEATHKIT COMPUTERS

System Engineered for Personal Computing

H8



Computers, peripherals and nearly 400 exciting, easy-to-build electronic kits, all in your

FREE Heathkit Catalog

HEATH
Schlumberger

Heath Company, Dept. 377-390
Benton Harbor, MI 49022

Please send me my FREE Heathkit Catalog.
I am not on your mailing list.

Name____________________

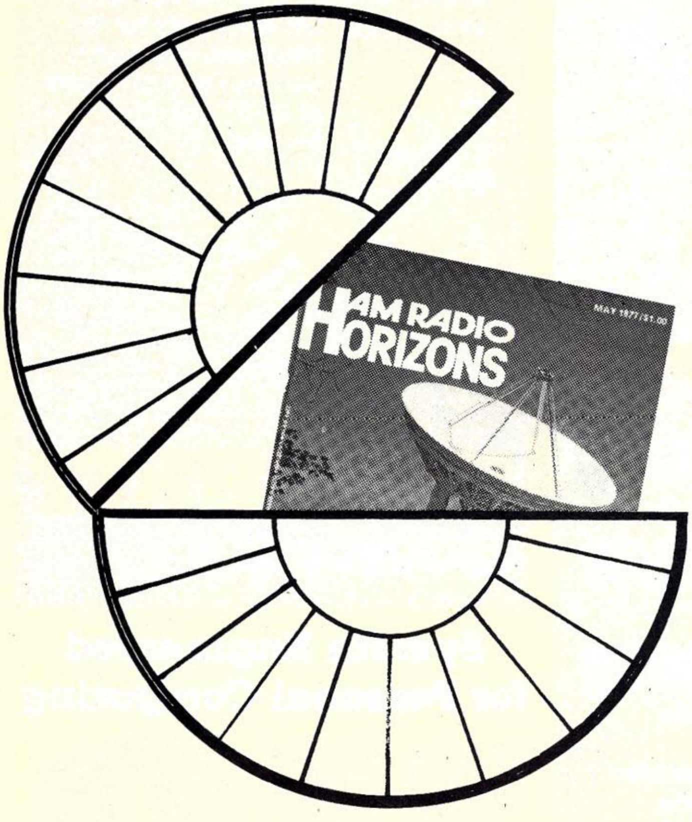
Address____________________

City____________ State________

CP-143 Zip________

for Openers...

Here's the ONE and ONLY Amateur Radio Magazine written for the Novice and Beginner



It's true. Only Ham Radio **HORIZONS** is edited specifically with *you* in mind! Each month the exciting articles in HORIZONS will examine the challenges you face as you get started in this fascinating hobby — setting up your station, improving your operating and upgrading your license. Ham Radio HORIZONS also features exciting special interest areas like contesting and DXing plus radio nostalgia fiction and humor. And all written from the Beginners viewpoint in a language you can thoroughly understand and enjoy! **EXPERIENCE** all that Amateur Radio can offer you — subscribe now and receive the next fascinating issue.

Pick up a copy at your local radio store or subscribe right now.

and a FREE LOOK at the whole scene...

from HAM RADIO'S COMMUNICATIONS BOOKSTORE — Amateur Radio's largest and most complete mail order bookstore. Our FREE 32-page book catalog describes hundreds of books and study aids you'll be looking for as you progress into ham radio. Practically every book relating to Amateur Radio from most major publishers is listed here. Send for your FREE copy today!

Spring
77
ham radio
BOOK catalog
ORDER TODAY TOLL FREE
800-258-5353

HAM RADIO HORIZONS

Greenville, NH 03048

☐ Please send FREE Book Catalog

☐ Please send a 1 year subscription to HORIZONS (12 issues) at $10.00

I enclose check or money order.

Name ______________________________

Address ______________________________

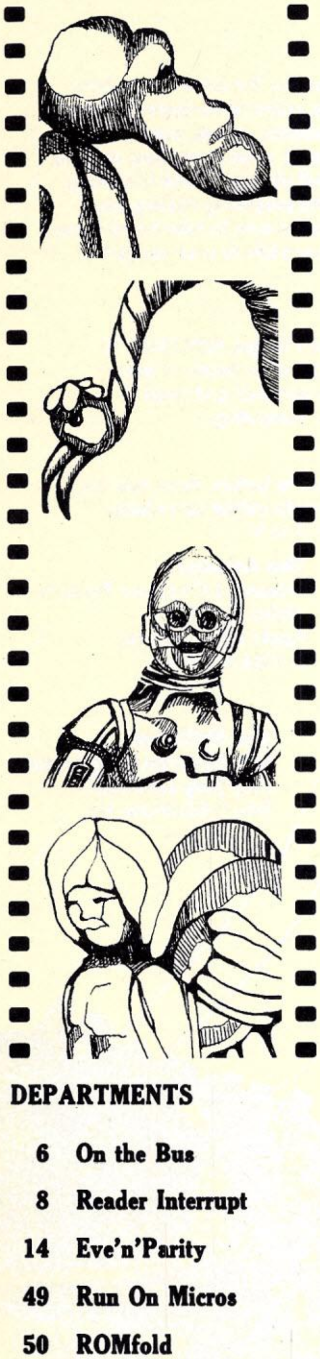
City ______________________________

State ______________ Zip ______________

Contents

Volume I, Number 9
March/April 1978

ROM
COMPUTER APPLICATIONS FOR LIVING



FEATURES

26 "Daddy, Is It the PET?" **by Richard Rosner**
One family's introduction to Commodore's PET computer—including the discovery of how to interface it with a printer for hard copy.

34 Incomprehensible Programs **by Joseph Weizenbaum**
Does anyone still remember how the program they wrote works? Who is really the author of the Pentagon's giant programs? Are man's ideas and concepts being crunched to fit the computer's perspective? If so, what perils does the future hold?

54 Time: Real and Unreal **by Sandra Erikson**
An introduction to the concept of real time—for computers—and some questions on the nature of time itself—for man.

58 The Absolute-Time Clock **by Lee Felsenstein**
The computer gets a new clock—and why it needs one to deal with the world beyond its hard shell. Complete with how-to-build-one-yourself instructions and schematics.

66 Things Computers Still **Can't Do** **by Hubert L. Dreyfus**
An incisive look into the problems facing the Artificial Intelligence community, including the image they've generated and have to live with.

71 Why Interpret? **by Eben Ostby**
An introduction to interpreters and their implications for computing. Also a brief look at LISP and its interpretation.

76 Othello **by Daniel Brodsky**
Othello was designed as a board game, but it's perfect for computers. Here's a championship Othello program that's hard to beat, and it beats any other program published to date, too.

86 The Oedipus Transaction (A short story) **by Robert Abel**
Cash flow—that's what it was all about. With the aid of his computer, J. D. "Steady" Stedmund could kite some $15,000 a minute. And his daughter was the best pupil he ever had.

DEPARTMENTS

6 On the Bus
8 Reader Interrupt
14 Eve'n'Parity
49 Run On Micros
50 ROMfold
83 Babbage and Lovelace
100 PROMpuzzle

COLUMNS

12 Missionary Position by Theodor Nelson
Out on the Language Limb

18 AIQuotient by A. I. Karshmer
ROM's Robot Review, Part II: Quasar—Fact, Fancy, or Fraud

22 The Human Factor by Andrew Singer
All Keyed Up

24 PROMqueries by Eben Ostby
Have a Question? Ask ROM

92 Cryptic Computer by Frederick W. Chesson
World War II's Legacy to Digital Techniques

98 FutuROMa by Bill Etra
The Cat Computer: Scratch, Purr, Fetch Data

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Domestic subscriptions are $15 for one year, $28 for two years, and $39 for three years. Canada and Mexico $17 for one year, $30 for two years, and $41 for three years. For European and South American subscriptions, please add $12 per year additional postage. For all other continents, please add $24 per year additional postage. *Copyright © 1978 by ROM Publications Corporation. All rights reserved.* Reproduction in any form or by any means of any portion of this periodical without the written consent of the publisher is strictly prohibited. The following trademarks are pending: AIQuotient, Babbage and Lovelace, Cryptic Computer, Eve'n'Parity, floppyROM, FutuROMa, The Human Factor, Legal ROMifications, Missionary Position, The Noisy Channel, On the Bus, PROMpuzzle, PROMqueries, Reader Interrupt, ROMdisk, ROMshelf, ROMtutorial, and Run On Micros. Opinions expressed by authors are not necessarily those of *ROM* magazine, its editors, staff, or employees. No warranties or guarantees explicit or implied are intended by publication. Application to mail at second-class rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

Everyone's getting personal in Long Beach.

3 full days of technical sessions, exhibits,
home-brew displays and the latest on personal and
small business computing, all at PERCOMP '78.
April 28-29-30.

Just for the fun of it, we have an entire home-brew section... robotics, games, computer music, even every-day, sensible stuff like checkbook balancing and preparing mailing lists. You're sure to take home some new tricks to your computer.

And don't forget, PERCOMP '78 has booth after booth of everything in personal and small business computing.

Jim Butterfield is on his way from Toronto with the entire, unabridged truth about KIM. Jim co-authored *The First Book of KIM.*

Attorney Kenneth Widelitz will be on hand with some friendly advice on "Tax Aspects of Lemonaide Stand Computing" while his friend attorney Leonard Tachner delivers the low-down on "Patents, Copyrights and Computers."

5 months before show time our dynamite exhibit list includes from A to V:

The Astute:
Advanced Computer Products
Alpha Supply Co.
Apple Computer, Inc.
A-Vidd Electronics

The Brilliant:
Byte Industries Incorporated
Byte Shop Lawndale
Byte Publications, Inc.

Carol Anne Ogdin's keynote address bares the facts on "How Personal Computers Are Being Used Today." Carol comes to us from Software Technique, Inc. in Alexandria, Virginia.

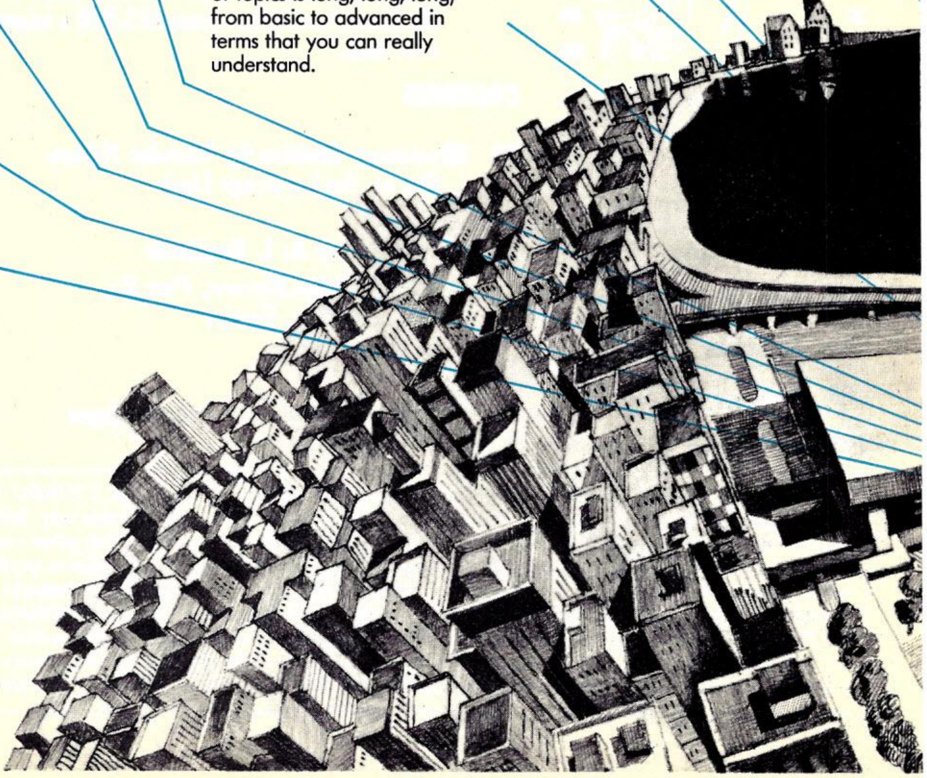
Admission for 3 full days of personal computing, complete with 180 exhibits, 66 fascinating seminars and all the going and coming you want is $10 ($8 for students and juniors) at the door, and $8 ($6 for students and juniors) if you pre-register.

Dr. Portia Isaacson, a contributing editor for Datamation and an associate of Byte, brings computer enthusiasts the very latest word on "Computer Store Retailing."

Whether you're just a beginner or a well informed expert, you'll find the latest on ham radio communications, graphic systems, word processing, pattern recognition or...(our list of topics is long, long, long) from basic to advanced in terms that you can really understand.

Louis Field, president of the International Computer Society/SCCS, gives you everything he's got on "Getting Started in Micro-Computing."

From Creative Computing Magazine comes David Ahl with all you'll ever need to know on "Marketing for the New Manufacturer."



RIP THIS COUPON FROM THE PAGE AND GET IT TO US BY APRIL 10.

I want to save time and money.
Please send me_______
pre-registration forms.

PERCOMP '78

1833 E. Seventeenth St., Suite 108, Santa Ana, Ca. 92701
Tel. (714) 973-0880

Name____
Address______________________________________
City____________________ State________ Zip_______
ROM-3

The Captivating:
Calcomp
Computalker Consultants
Computerland
Computer Magazine
The Computer Mart of Orange
Computer Pathways Unlimited, Inc.
Computer Power & Light Inc.
Creative Computing

The Dynamic:
Databyte, Inc.
D.C. Hayes Assoc.
Dilithium Press
Dynabyte, Inc.

The Energetic:
Edwards Assoc.
Electronics Warehouse, Inc.
Electro-Sonic Components, Inc.
Entech

The Hearty:
Heathkit Electronic Centers
Hobby World

The Irresistible:
Interface Age Magazine

The Jovial:
Jade Company
James Henry Co.

The Keen
Kathryn Atwood Enterprises
Kilobaud Magazine

The Magnificent:
Marinchip Systems
Micropolis Corporation
Mission Control

The Omnipotent:
OK Machine & Tool Corp.
Olson Electronics, Inc.
Optical Electronics, Inc.
Orange County Computer Center

The Personable:
Pan Dynamics, Inc.
Personal Computing
Problem Solver Systems, Inc.

The Quintessential:
Quainco Ltd.
Quest Electronics

The Remarkable:
Radio Shack
ROM Publications, Corp.

The Sterling:
SD Sales
Space Byte Corp.
SubLogic Co.
Sunshine Computer Co.
Sybex, Inc.
Szerlip Enterprises

The Tantalizing:
Tandy Computers
Tarbell Electronics
Tech-Mart
Telpar, Inc.
TLF, Corp.

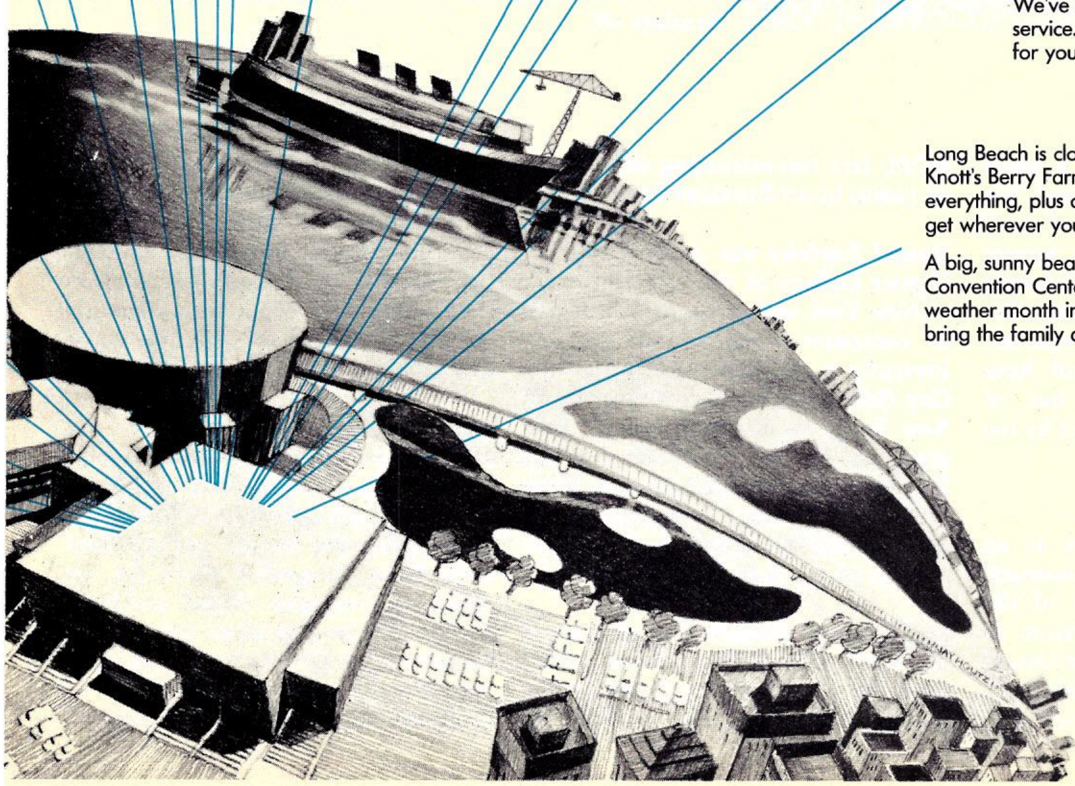
The Ultra:
Ultra-Violet Products, Inc.

The Valiant:
Vector Graphics, Inc.
Vista Computer Co.

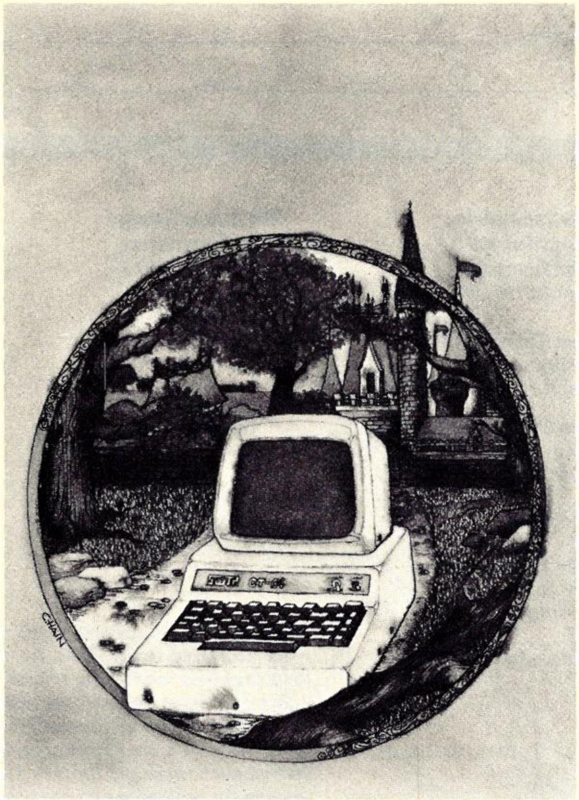
Since everybody's coming, better make your advanced reservations. Pre-register and save (you won't have to wait in line) ...but don't forget about your hotel room. Our staff has reserved rooms in hotels and motels near the Convention Center. We've even arranged for a shuttle bus service. So call and we'll save a room for you.

Long Beach is close to Disneyland, Knott's Berry Farm, Universal Studios... everything, plus our staff will help you get wherever you want to go.

A big, sunny beach is minutes from the Convention Center, and April is a great weather month in Long Beach, so plan to bring the family and have a good time.



On the Bus



Cindy Hain is a recent graduate of Parson's School of Design who has been a Contributing Artist at *ROM* since the beginning. She enjoys designing needlework as well as working with pen and ink or watercolor. Her favorite artists are Paul Giovanopoulos and Andrew Wyeth. What she wants most in the world now, she says, is to have Saturdays and Sundays off.

Robert Abel is a journalist, a small press editor (Lynx House), and has published fiction in a number of small press journals, including *Epoch, Kansas Quarterly, Dark Horse,* and *Colorado State Review*. He is a closet science fiction writer. A former college teacher, he sometimes thinks he would have been better off as a jockey, but, of course, all that has been ruined by too many beers.

Hooked on crossword puzzles at an early age, **Daniel Alber** now constructs as well as solves them. Part of the brownstone renovation generation in New York, when he's not constructing puzzles for the likes of *Field and Stream, The New York Times,* and *ROM,* he's reconstructing olden golden rooms in his Brooklyn-based house.

Daniel Brodsky was graduated from Hunter College of the City University of New York with a bachelor's degree in computer science and geology. Presently a graduate student at the City College of the City University of New York, he is also a systems programmer in the Technical Services Division at the Federal Reserve Bank of New York, where he maintains system software and handles system generation on the Burroughs B7700.

Frederick W. Chesson is a graduate of the University of Connecticut. After work in electronic engineering, he gravitated into technical writing. At present, he furnishes instruction manuals and related items to various firms plus construction articles to several electronics hobby magazines. A member of the American Cryptogram Association since 1958, he is currently researching a book on Civil War codes and ciphers.

Hubert L. Dreyfus, who received his Ph.D. from Harvard, has taught at M.I.T. and the University of California at Berkeley. His revised and updated book *What Computers Can't Do,* on which this month's *ROM* article is based, will be published this spring. Currently, Dr. Dreyfus is involved in research on the use and limitations of computer models in the social sciences.

Sandra Erikson is an independent consultant specializing in commercial/agricultural lighting. As a free-lance writer, her articles have appeared in a number of national magazines, including *New York.* Her hobbies include tending the greenhouse and making plum brandy.

Bill Etra is a West Coast-based computer design consultant. He is coinventor of the Rutt/Etra Video Synthesizer—the first portable voltage-control analog video synthesizer, as well as the Videolab. His main interest is videographics, and many of his works have appeared as cover illustrations on various periodicals and books including *Computers in Society* and *Broadcast Management and Engineering.* His current research centers on "The Computer as a Compositional Tool for Video."

Lee Felsenstein was born in Philadelphia and grew up wanting to be an inventor. Outside of that, he bears no resemblance to W. C. Fields whatsoever. Instrumental in establishing the first experimental public-access information-exchange system in 1972, he is presently engaged in further development in that area of communications. In his spare time he has designed the Pennywhistle 103 modem, the VDM-1 video display module, the Sol terminal/computer, and the VID-80 video display card. Lee was also instrumental in forming the original Homebrew Computer Club and currently serves as its "toastmaster."

A. I. Karshmer is currently completing his Ph.D. in computer science at the University of Massachusetts. His main interest is the use of artificial intelligence concepts in solving problems involved in the transmission of computer graphics. Currently, he is developing a method for sending high-density information, such as animated graphics, over existing low-bandwidth telecommunications networks.

Theodor Nelson is the author of the classic *Computer Lib/Dream Machines,* a Whole-Earth-style catalogue of computer machinations. His latest book is the newly released *The Home Computer Revolution.* Ted specializes in highly interactive systems for graphics and text. His past experience includes a stint at Dr. Lilly's Dolphin Laboratory and work as a consultant for Bell Lab's ABM system.

Eben Ostby has been involved with computing ever since he crashed the PDP-8 at Pomfret School. At present, he is doing graduate work in computer science at Brown University and trying to convince people that APL isn't really all *that* bad.

Richard Rosner is a Project Engineer and systems designer for the Perkin-Elmer Corporation. His specialty is microcomputer-based intelligent management terminals and graphics. Computer articles penned by Rosner have appeared in *EDN, Electronics,* and *BYTE.*

Andrew Singer has been hooked on computers since he first built one in 1958. A hard/software consultant, he is fluent in thirty computer languages and knows more than enough about twenty species of machine. His work has included the first medical information retrieval system based on ordinary clinical records, and a large and intricate system for interactive selection of data from public opinion polls. He believes that most software is poorly designed and unspeakably rude, and his Ph.D. research is aimed at improving the architecture and human engineering of interactive systems.

Joseph Weizenbaum is Professor of Computer Science at the Massachusetts Institute of Technology. He is best known to his colleagues as the composer of SLIP, a list-processing computer language, and for ELIZA, a natural-language processing system. More recently, he has directed his attention to the impact of science and technology—and of the computer in particular—on society.

A Note to Authors:

ROM is always looking for good computer applications articles from people with up-and-running systems. We also will be glad to consider for possible publication manuscripts, drawings, and photographs on other computer-related subjects. Manuscripts should be typewritten double-spaced, and a stamped self-addressed envelope of the appropriate size should accompany each unsolicited submission. Although we cannot assume responsibility for loss or damage, all material will be treated with care while in our hands. Manuscripts should be sent to ROM Publications Corporation, Route 97, Hampton, CT 06247.

COMPUTER MART
OF NEW YORK, INC.

"THE MILLION-DOLLAR STORE THAT STILL SELLS 25-CENT CHIPS WITH A SMILE."

It doesn't matter if you're looking for a $10,000 system or a 10-cent diode. You'll get the same friendly, helpful service at Computer Mart of New York.

• • •

CHECK OUT THE NEW SPECIAL

8K Static Board built to Computer Mart's exact specifications. Complete and running. Fully guaranteed. Only $175.00.

118 Madison Avenue
New York, New York 10016
212-686-7923

RAINBOW COMPUTING, INC.

Supplier of

WAVE MATE
THE DIGITAL GROUP
DEC PDP
Computer products

Peripherals and Supplies from

PERSCI
CENTRONIX
DIABLO
MAXELL
COMPUTER DEVICES
LEAR-SIEGLER
MULTI-TECH
TEXAS INSTRUMENTS

Specialists in Design, Implementation
and Support of
Custom Hardware/Software Systems for
Business, Educational, and Personal Use

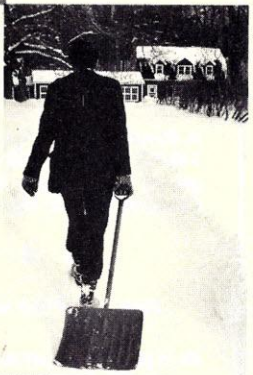
Experts in most major computer
software including
CDC, IBM, PDP
BASIC, COBOL, FORTRAN, PL1
LISP, SIMULA, SNOBOL, SPSS, BMD's
COMPASS, MACRO,
6800, & Z80 assembly languages

10723 White Oak Ave., Granada Hills,
Ca. 91344
(213) 360-2171

Reader Interrupt

The road to ROM is paved with good intentions and lots of snow.

This is one job that should *be replaced by a sturdy computer.*



The horse-drawn S-100 sleigh on our January cover proved to be much more prophetic than we expected—indeed much more prophetic than we liked. The issue was snowed in from beginning to end. First came a blizzard here in Connecticut. Then it was delayed by an ice-storm-induced power failure. Finally the Columbus Bank Note division of Banta was socked in by Ohio's worst blizzard of the century. (They used to print money, so while they were snowbound and waiting, we tried to convince them to replace our regular centerfold with leftover $100 bills from the old days—no luck, unfortunately.)

When the presses were running again, the trucks weren't. The issue idled, awaited binding. Then the ice slid the big delivery trucks to the sidelines.

All in all, it was enough to make us wish for sleighs. Actually we have six of them in the barn next to the ROM *office—but no horses. And there are some things computers just can't do.*

ROMulus

LATE BLIZZARD FLASH

Just as we finished setting type for this issue, we were hit by a blizzard that made the last one look like a sunny day in Silicon Valley. The Army was airlifted in to help clean up the mess. Even so, we were snowed in for a week until a big bulldozer with a twelve-foot-wide blade crawled in to rescue ROM. With nothing but cabbage soup and cabin fever to keep us busy, we reluctantly tore off all the folios in this issue and replaced March *with* March/April. *Without combining issues, it probably would have taken all year to get back on stream.*

Subscribers, please note, you'll still receive the twelve, twenty-four, or thirty-six issues you signed up for. You won't be shortchanged. As for lifetime subscribers—what can we say except that we hope you live a month longer than fate had in store!

Many thanks for your patience, and keep those letters and calls coming—but please, even though we've purchased six new snow shovels, don't send snow. And, by the way, we intend to have a steaming jungle picture on the January 1979 cover!

ROMulus

Dear ROM,

Enclosed is a copy of a "Proposal for Computer Assisted Bible Study." As you can see, it is going to be a very

All right, so we were *dreaming of a white Christmas, but do you think we could con Santa into delivering our ROMs?*

Gottcha.

The new Pup-1 from Seals Electronics arrives in the middle of the blizzard, doing more to warm up Tanya than the glowing box stove which heats ROM's art department could ever do.

Photographs by Thomas Hall

Every computer needs a ROM - so do you!

ROM
COMPUTER APPLICATIONS FOR LIVING
SOL: THE INSIDE STORY by Lee Felsenstein
ALTAIR AND THE ART OF MOTORCYCLE SHOP MAINTENANCE
TOOLS by Joseph Weizenbaum
MEMORY, MEMORY, HOW MUCH MEMORY? by Stan Veit
APLomania by Eben F. Ostby
BINARY CLOCKS IN 0011/0100 TIME
BIOFEEDFORWARD by Bill Etra
COMPUTER WRESTLING: THE PROGRAM OF CHAMPIONS
XEROXES & OTHER HARD COPY OFF YOUR CRT
A PAYROLL PROGRAM FOR YOUR SMALL BUSINESS
THE WONDERFUL DREAMS OF DR. K
SCOTT JOPLIN ON YOUR SCI-FI HI-FI
KILOBYTE CARD Memory for Pennies
8080
PROJECT PROMETHEUS by Lee Felsenstein
HOW DOES YOUR CRYSTAL GROW by Sandra Faye
MAKE ME MORE MUSIC, MAESTRO MICRO by Dorothy Siegel
THE WORDSLINGER 2200 Characters Per Second

Computers Challenge America's Cup by Eben Ostby ☐ *A Beginner's Guide to Peripherals: Input/Output Devices Your Mother Never Told You About* by Leslie Solomon and Stanley Veit ☐ *Computer Country: An Electronic Jungle Gym for Kids* by Lee Felsenstein ☐ *The Best Slot Machine Game Ever* by Tom Digate ☐ *The Micro Diet: Better Health through Electronics* by Karen E. Brothers and Louise L. Silver ☐ *Come Closer and We Won't Even Have to Talk* by Avery Johnson ☐ *The Kit and I, Part Four: Testing, Testing* by Richard W. Langer ☐ *Computer Models in Psychology* by Joseph Weizenbaum ☐ *Micro, Micro on the Wall, How Will I Look When I Am Tall?* by Stuart Dambrot ☐ *Copycat Computer* by Tom Digate ☐ *Talk Is Cheap* by Hesh Wiener ☐ *Project Prometheus: Going Solar with Your Micro* by Lee Felsenstein ☐ *BASIC from the Word GOTO* by Eben Ostby ☐ *Chipmaker, Chipmaker, How Does Your Crystal Grow?* by Sandra Faye ☐ *The Kit and I, Part Three: Personality Plus* by Richard W. Langer ☐ *Make Me More Music, Maestro Micro* by Dorothy Siegel ☐ *Wings in Wind Tunnels: Computer Models and Theories* by Joseph Weizenbaum ☐ *What Is a Microcomputer System?* by Leslie Solomon and Stanley Veit ☐ *Maintaining Your Micro* by O.S. (The Old Soldier) ☐ *Time Sharing on the Family Micro* by Barry Yarkon ☐ *The Wordslinger: 2200 Characters per Second* by Stuart Dambrot ☐ *Light Fantastic: The Kinetic Sculpture of Michael Mayock* by Tom Moldvay and Lawrence Schick ☐ *From Bombs to ROMs* by Lavinia Dimond ☐ *Guard against Crib Death with Your Micro* by Jon Glick ☐ *Home Computers: The Products America May Never Know It Needs* by Martin Himmelfarb ☐ *Putting Two and Two Together* by Tom Pittman ☐ *The Wonderful Dreams of Dr. K* by Hesh Wiener ☐ *The Kilobyte Card: Memories for Pennies* by Thorn Veblen ☐ *The Unlikely Birth of a Computer Artist* by Richard Helmick ☐ *Scott Joplin on Your Sci-Fi Hi-Fi* by Dorothy Siegel ☐ *Building a Basic Music Board* by Eben F. Ostby ☐ *The Compulsive Programmer* by Joseph Weizenbaum ☐ *The Very Best Defense (a short story)* by Laurence M. Janifer ☐ *Chart Up and Flow Right* by Eben F. Ostby ☐ *Computer Wrestling: The Program of Champions* by Lee Felsenstein ☐ *Forget Me, Forget Me Not* by Avery Johnson ☐ *PLATO Makes Learning Mickey Mouse* by Elisabeth R. Lyman ☐ *Charged Couples* by Sandra Faye Carroll ☐ *Xeroxes and Other Hard Copy off Your CRT* by Bill Etra ☐ *The Kit and I, Part Two: or Power to the Computer* by Richard W. Langer ☐ *How Computers Work* by Joseph Weizenbaum ☐ *Personally Yours from IBM* by Eben F. Ostby ☐ *A Payroll Program for Your Small Business* by Robert G. Forbes ☐ *Memories Are Made of This* by Lee Felsenstein ☐ *Memory, Memory, How Much Memory?* by Stan Veit ☐ *Software—The Genie in the Bottle*

The computer magazine for the curious

Every monthly issue keeps you abreast of the latest microcomputer applications for home, school, and office. Written by professionals who know how to present microcomputing in a lively, readable, and understandable fashion, ROM is fun. ROM is instructive. ROM is everything you ever wanted in a computer magazine.

Look what you've been missing without your monthly ROM!

Plus columns by Ted Nelson, Andrew Singer, Bill Etra, and A.I. Karshmer, on Artificial Intelligence, The Future, The Human Factor in Computing.... Plus practical software, listings, documentation, new peripherals, interfaces, games.... And more, much more.

ROM
COMPUTER APPLICATIONS FOR LIVING

ROM Publications Corp.
Route 97, Box R
Hampton, CT 06247

Name ____________________

Address ____________________

City __________ **State** __________ **Zip** __________

U.S.A.: ☐**One year $15** ☐**Two years $28** ☐**Three years $39**
Canada & Mexico: Please add $2/yr. additional postage
Europe & South America: Please add $12/yr. additional postage
All other continents: Please add $24/yr. additional postage
☐**Check/money order encl.** ☐**Master Charge** ☐**BankAmericard**
Exp. date __________ **Card#** __________

Please allow 4-6 weeks for delivery.

large undertaking and I could use some help, which is why I'm writing to ROM and its readers.

Since the Bible contains about 3.5 million characters, it obviously would be a large data entry task to translate it into computer format. Help from others in mutual sharing of the effort to place the KJV into computer format would be greatly appreciated. Possibly later other versions could also be computerized.

Also needed will be a text editing-word processing program to handle the storage, retrieval, and updating of reference notes relating to the various Bible verses. The ability to cross reference and display verses which contain related information is desired and will be a task to program which possibly could be better handled by a mutual sharing of effort. There could be a problem of incompatibility between systems to be worked out.

For the purpose of cross referencing, it is suggested that the book, *Treasury of Scripture Knowledge* which consists of 500,000 Scripture references and parallel passages, also be keyed into computer format for storage on computer tape or disk for reference purposes. This book is available at most Bible bookstores for less than $10.

Once the data base and the programming is established, it should be possible to use it to cross check references, and comments to see if there are any Bible verses to support or to oppose the subject under consideration. Also a check can be made of any comments already entered into the data base regarding an explanation of the reference under consideration.

Another possible use could be the use of the original Greek and Hebrew texts to check out translation accuracy and other problems.

If you are interested in taking part in this effort, or know of someone who would be interested, or if you know where some of the above goals have already been accomplished, please contact me. Thank you.

Larry E. Ellison
19 Huntington Lane
Willingboro, New Jersey 08046

Dear ROM,

Having read the Gat story, "Conversion of the Diggers," I say *ROM* has found what should become a programmed feature—a Gat story each month.

His stories are serious, well-written, and show he knows about technical equipment.

Can we *please* have another short story by Gat.

Gregory C. Wilson
Florence, Massachusetts

Coming up soon.

ROMulus

Dear ROM,

It's nice of you to answer readers' questions in your magazine, but your readers should know that for simple questions (definitions, acronyms, basic explanations) there is a quicker, closer source—the public library.

The reference librarian at most public libraries is eager and waiting to answer your questions, in person or by phone. If a reader prefers to search for answers himself, he should find one of the following sources, or something like them, in the library's reference section:

Computer Dictionary by Sippl & Sippl (Howard J. Sams & Co., 1974).

Computer Dictionary by Donald Spencer (Camelot Publishing Co., 1977).

Encyclopedia of Computer Science, edited by Ralston and Meek (Petrocelli/Charter, 1976).

If a local library doesn't have sources like these, it will appreciate the recommendation that it buy them! The dictionaries are paperbacks costing under $10; the encyclopedia costs $60.

A novice in the computer field might consider buying one of the dictionaries for him/herself, or buying any of the numerous "computer fundamentals" books with a good glossary. Then he/she can save the really tough, interesting questions to send to PROMqueries!

Please pass this information on to your readers!

Margery Goldstein
Technical Librarian
Digital Equipment Corp.
Maynard, Massachusetts

Consider it done. Meanwhile keep those questions coming. We'll be glad to answer them.

ROMulus

ROM
COMPUTER APPLICATIONS FOR LIVING

Editor and Publisher
Erik Sandberg-Diment

West Coast Editor
Lee Felsenstein

Associate Editor
Sue Neillson

Assistant Editor
Lynn A. Archer

Contributing Editors
Frederick W. Chesson
Bill Etra
Louise Etra
Sandra Faye
Ed Hershberger
Avery Johnson
Arthur Karshmer
Richard W. Langer
Theodor Nelson
Robert Osband
Eben Ostby
Frederik Pohl
Andrew Singer
Alvin Toffler

Crossword Puzzle Editor
Daniel Alber

Editorial Assistant
Donna Parson

Art Director
Susan Reid

Assistant Art Directors
Cindy Hain
Korkie

Staff Photographer
Thomas Hall

Contributing Artists
Steve Gerling
Robert Grossman
Luis Jimenez
Rex Ruden
Linda Smythe

Circulation Director
Jennifer L. Burr

Counsel
Peter Feilbogen

Dear ROM Ms,

We are in the midst of a computer revolution, and women have a chance to get involved in this new science, art, discipline, hobby. We women have a chance to open up a new world for ourselves. The men aren't trying to keep us out of this one—we should get involved!

What's that? Computers are too technical for women? You need too much math? It's too hard? Wrong! Wrong! Wrong! Computers are fascinating, fun, and a real challenge for your mind.

Who am I, you ask. Well, I'm a former chorus girl (*My Fair Lady, How To Succeed In Business*. . .). I'm now a wife, mother, and novice in the world of computers. I have no technical background; I left college to pursue a career as a dancer. The discipline I learned as a dancer will come in handy when I deal with computers. You, too, may be able to relate some skill you already possess in the same way. My background isn't a fascinating subject, but simply a way of illustrating my point—women can and should become involved with computers. It's really not that difficult.

If you're considering entering the job market out of necessity, or because you need to be more than a homemaker, or you're still in school and deciding on a career, think computers. Perhaps you want something to do while the kids are in school—a hobby—think computers. Computers can offer you the opportunity of a lifetime—good pay and interesting work.

Is your husband, lover, son, etc., a computer hobbyist? You can share this interest with him. Don't let the men have all the fun.

Where should you start? Begin by reading this magazine. It's really informative. Read *all* the hobbyist magazines. When the guys go to the computer store, go with them. Or better yet, go by yourself. Don't be ashamed to ASK QUESTIONS! (When you were a new mother, didn't you learn by asking questions and from experience? The same techniques can be used to learn about computers.) Men have been known to ask dumb questions too. To help answer some of your questions, I would definitely recommend reading Ted Nelson's book *Computer Lib*. It's an easy book to read and full of information.

Girls, our world is changing. Our role is changing. Computers are going to have a tremendous influence on our lives. Be a part of it. Don't say, "It's too technical for me." You have a mind—use it.

In the very near future, you'll probably have a computer in your home. Your children will surely have computer-aided instruction in school. A computer is a machine just like a typewriter or vacuum cleaner. It won't eat you up. It'll only do what *you* tell it to do.

I'm not pushing computers as a feminist, but rather as a person who finds herself in an exciting, new world. I just want to share my enthusiasm with you.

Marilyn Karp
Briarwood, New York

P. S.

To all of you ROM men out there: If your woman is now showing an interest in the subject of computers, encourage her. Give her MORAL SUPPORT. Answer her questions and you will reap the rewards.

How Well Are You EATING?

Are you curious about the nutritive content of your favorite budget casserole? Wondering if that new recipe will fit into your special diet? Or just want to know if that delicious dessert had any "redeeming" nutritional value? Team up your computer with NUTRIVALUE* and get answers!

The NUTRIVALUE personal nutrition analysis programs allow you to analyze recipes, meal plans, and daily or weekly menus on your home computer. Just type in the list of ingredients; your computer, running NUTRIVALUE software, will compute and print (or display) the analysis. NUTRIVALUE comes in two versions; pick the one that suits your configuration:

NUTRIVALUE I

- Analyzes 12 nutrients
- Numerically coded input
- Contains nutrient data for 53 food items; user expandable
- Tabular output format
- Written in BASIC; does not require string functions or file manipulation
- Source program requires 5K bytes of memory; user reducible

NUTRIVALUE II

- Analyzes 17 nutrients
- Ingredient specifications can be entered by name
- Choice of 100-food item or 200-food item data base
- Tabular output format
- Written in BASIC; requires string functions and file manipulation
- Source program requires 5K bytes of memory, user reducible, and file-structured secondary storage

* NUTRIVALUE is a trademark of Consultus

For more detailed information about
the NUTRIVALUE programs,
send this coupon to:

Consultus
P. O. Box 86
Arlington, MA 02174

Please send me more information about NUTRIVALUE

Name ____________________

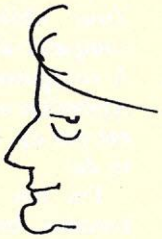
Address ____________________

City __________ State __________ Zip __________

Computer configuration ____________________

Missionary Position

OUT ON THE LANGUAGE LIMB



by
Theodor
Nelson

Computer Languages

The original computer languages were simplified ways of writing down the undermost small instructions of the computer, one by one. This is still very respectable. But when such details do not matter in their particulars, we use programming systems that take care of such things for us, letting us think clearly about the instructions that really need our attention. These systems are often called "higher" languages, or just computer languages.

Surprise! There are thousands of different computer languages.

There are at least two dozen important computer languages; the experienced programmer generally knows between three and seven.

The Higher Languages

A computer is, as we have seen, a device for following a plan. This plan can be expressed in any number of ways, provided that the computer is properly set up to recognize and carry out the steps of the plan. Computer languages are simply these different ways of expressing plans. And there is no single standardized way.

The different computer languages arise from the profusion of things computers can do. Computers can do so many things—pictures and music and printing and sorting, not to mention numerical applications—and the more you think about it, the more different possible things you may want the computer to do.

There are many kinds of things people want done with computers, and many styles for doing them. Indeed, little astonishes the newcomer as much as the complete blankness of the computer, the fact that it really can be made to do anything whatever that its electronics will allow.

But different people have different things in mind. Since the very beginnings, many have used the computer for rapid numerical calculation. Others use the computer principally for business accounting and for storing records of business transactions. Yet others see the computer as an extremely deft motion-picture toy.

All these people are right; no one is wrong. But with these different emphases, and the natural variation of human mentality, many different styles of programming, and local rules of operation for programmers to follow, have come into being. By and by, using the computer for a given range of problems, and in a certain style, gives rise to a new programming language. A computer language does not jump out of the air. It is designed by someone to be a useful way of telling the computer what he or she wants.

Each of the higher computer languages allows you, as a rule, to program some particular range of problems, and in a particular style. In part this is because each language handles a lot of details for you automatically. Today's larger programs call in dozens or thousands of littler programs which have themselves been perfected—little programs for putting things in alphabetical order, typing a character on a terminal, moving a picture on a screen, and thousands of other functions. These are called *subprograms* and are of various types. While you do not want to have to create each of these subprograms, you want to be able to use them. So you need a shorthand method of telling the computer to carry out these little programs, and of tying them together. And such a shorthand method is a computer language.

Beginners are startled to learn what a lot of different computing languages there are, and what little agreement about their merit there is among experts. Indeed, laymen commonly ask "How do you say it in computer language?" and this has no general answer at all—because there are so many.

Just as the blind men misconstrue the elephant, and just as different computer users see the computer differently, different computer users likewise prefer different languages, because the different languages are tied to people's different ways of seeing and areas of concern.

People get very uptight about computing languages; the subject is as touchy as religion, if not more so. If you insult a man's favorite computer language, you cease to be his friend.

Indeed, there is no more emotional issue in the computer field than that of computer languages. While physical violence rarely occurs, the levels of emotional commitment and rage to be seen when computer people discuss computer languages is truly awesome. Many hobbyists who have only learned BASIC tend to go through this stage. Since all they have seen are programs in BASIC, all they can imagine is programs in BASIC, and thus they naturally think computers can have no uses *except* those which are easily programmed in BASIC. And indeed they get indignant, just like regular computer people, to hear anyone say they might be missing something.

The most important subject for the computer beginner is not electronics or mathematics; it is a subject that did not in any way exist thirty years ago. It is the subject of computer languages.

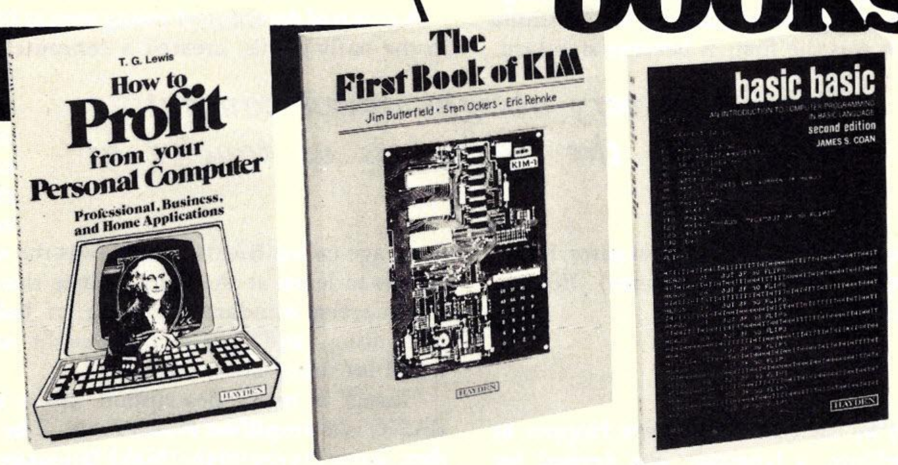
The Main Computer Languages

While this is not the place to get into computer languages deeply, let's at least do a rundown of some main areas. Because there are thousands of computer languages, there are also many different ways of categorizing them, . . . and the categorization we will make here is a simple categorization. (It might startle some professionals.)

From *The Home Computer Revolution,* Copyright © 1977 by Theodor Nelson.

New from Hayden!

#1 in personal computing books!


T. G. Lewis
How to Profit from your Personal Computer
Professional, Business, and Home Applications
HAYDEN
The First Book of KIM
Jim Butterfield • Stan Ockers • Eric Rehnke
KIM-1
HAYDEN
basic basic
second edition
JAMES S. COAN
HAYDEN

HOW TO PROFIT FROM YOUR PERSONAL COMPUTER:
Professional, Business, and Home Applications
By T. G. Lewis. Describes how to put the computer to work for you.
#5761-X, paper, 256 pp., $7.95

HOME COMPUTER SYSTEMS HANDBOOK
By Sol Libes. A technical look at personal computers.
#5678-8, paper, Available April, 1978

THE FIRST BOOK OF *KIM*
By Jim Butterfield, Stan Ockers, and Eric Rehnke. How to write *KIM* programs, with illustrations.
#5119-0, paper, 176 pp., $9.00

***FORTRAN* WITH STYLE: Programming Proverbs**
By Henry F. Ledgard and Louis J. Chmura. Programming style guide that conforms to the new definition of standard *FORTRAN*.
#5682-6, paper, 176 pp., Available May, 1978

BASIC *BASIC*, Second Edition
By James S. Coan. Fundamentals of *BASIC* programming.
#5106-9, paper, $8.95; #5107-7, cloth, 288 pp., $9.95

HOW TO BUILD A COMPUTER-CONTROLLED ROBOT
By Tod Loofbourrow. Provides an application of a microprocessor and hands-on experience with robotics. *#5681-8, paper, Available May, 1978*

These NEW books join our winning list of best sellers including:

Advanced BASIC (Coan), **The BASIC Workbook** (Schoman), **Game Playing with BASIC** (Spencer), **Digital Troubleshooting** (Gasperini), **Digital Experiments** (Gasperini), **Standard Dictionary of Computers and Information Processing** (Weik), **Telephone Accessories You Can Build** (Gilder)

Hayden Book
Company, Inc.
50 Essex Street
Rochelle Park, NJ 07662

Available at your local computer store!

Traditional Languages

In lumping together the following as "traditional" languages, I am taking a few liberties. . . . Traditional languages require the programmer to figure out ahead of time the exact division of memory to be used for each piece of information that needs to be stored or operated upon. One way or another, the programmer sets places aside for each kind or piece of information that will be needed. (This is one of the main pitfalls of the traditional languages, as it reduces their flexibility.)

FORTRAN

Because the first use of the computers was for arithmetical and formula computations, it was natural that a computer language should be developed which simplified the programming of algebraic formulas. This language was called FORTRAN, supposedly standing for "formula translation." Because it was the first, it became standard. Once it was a milestone; now it is a millstone. People learn it first because it is standard. It was originally designed for mathematical applications; but it is, in most cases, far inferior for these purposes to APL (described later). But still they go on teaching it in the universities.

> *People get very uptight about computing languages; the subject is as touchy as religion.*

COBOL

Spurred particularly by the efforts of Grace Hopper at the Department of Defense, a language was devised for business application, called COBOL (Common Business-Oriented Language). It has certain strengths, but is very inflexible compared to the Lambda languages (described later). COBOL programmers are the coolies of the computer field.

ALGOL

In Europe, mathematicians and scientists who became disturbed at the inflexibility of FORTRAN created a language capable of expressing (and thus programming) much more elaborate and subtle types of procedures. The resulting language, ALGOL, is widely used in other countries, and is standard even in this country as a way of writing down computing procedures so that other programmers can use and understand them. This is because it has no extraneous features, as does FORTRAN.

PL/I

The language PL/I (Programming Language I) was developed as an IBM product. Roughly speaking, it is a combination of FORTRAN, COBOL, and ALGOL all together, preserving the complications of each and the distinct philosophy of none. Many companies with IBM computers use it, however.

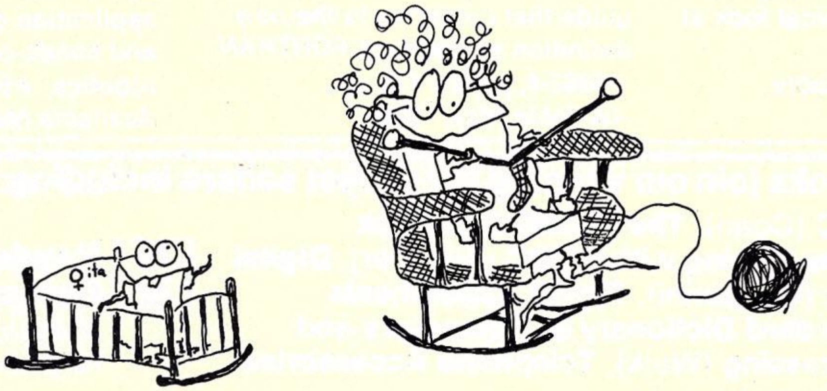
BASIC

A group of determined young men at Dartmouth College, in the early 1960s, created a computer system for everybody there to use, acting on the determination to make computers easy. For this they created a new programming language called BASIC, which was the simplest of all languages to learn at the outset. Since that time, BASIC has become the standard language of hobby and amateur computing, and indeed has caught on throughout the world for many other purposes.

"Basic" is not a description, it's a name. Essentially, BASIC is a simplified FORTRAN. The BASIC language, then, is not (as you might think) language somehow intrinsic to computers, but a language which was created to make programming quick and easy.

The fact that BASIC is easy to use does not mean it is efficient, and there are a lot of things that simply cannot be done in BASIC. Truly complex programs can be created in BASIC only with the greatest difficulty. However, the new computers being set up for home use all come with BASIC, and so its use is growing dramatically even while its limitations are felt ever more painfully by those concerned with creating really versatile and complex programs.

EVE'N'PARITY



"Knit 0001, purl 0010."

Cartoon by Korkie

By common consent, the amateur world is deeply committed to BASIC; but there is no exact standard of what BASIC is, and so there is plenty of room for improvement. One possible hope is that the best elements of LOGO could be slyly introduced to BASIC, until BASIC comes more and more to have some of the power of LOGO. (One sort of superBASIC, called GRASS, may become available soon for amateur machines.)

The Lambda Languages

The second category of computer languages will be, in the opinion of the author, the important ones for tomorrow. They offer a power, and, in some cases, a simplicity that has not been widely seen as yet. The Lambda languages are called that because they are based, somewhere deep down, on something called the Lambda Calculus. But you don't have to know about that.

This mysterious thing, the Lambda Calculus, is simply a systematic way of tying things together; of taking the results of one operation and making them the starting point of another operation. The Lambda languages, accordingly, are extremely versatile, as the results of any operation can be used as the beginning of any new operation. Thus, they have few of the restrictions that are so common in the other languages. Space need not be exactly prearranged, as in the traditional languages.

The Lambda languages were first used in obscure research laboratories, especially those where many delightful odd people work on what is referred to as artificial intelligence. The original Lambda language is called LISP, and it is so intricate and obscure to most computer people that its practitioners have come to be seen as strange eccentrics — a priesthood within the priesthood. Yet there was a reason for this strange computer language, and all of its frightening parentheses: anything which can be done in any other computer language can be done in LISP, while things that can be done in LISP cannot be done in any other computer language.

People versed in FORTRAN and COBOL were alarmed by LISP, because it contained hundreds of parentheses. The parenthesis is the most common character in LISP. This annoys and offends those who don't understand it, because they naturally think anything can be programmed in FORTRAN and COBOL, which is not true.

But LISP ordinarily only runs on big machines (although a group at MIT is endeavoring to build a LISP machine small enough to be a personal computer).

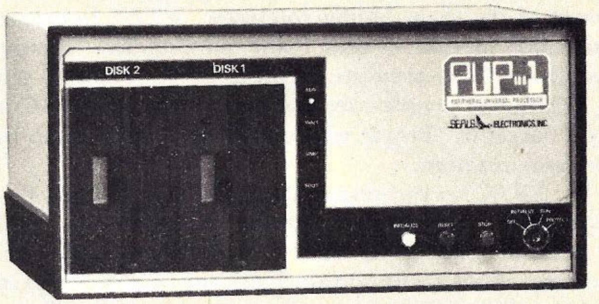
There are, however, other languages which have all the power of LISP, and yet have certain other advantages. An important one of these is LOGO. Created by Papert, Feurzeig, and others, LOGO is as simple to use as BASIC, but far more powerful. It may well become available for hobbyist computer machines in the near future.

A group at MIT, doing research in LOGO as a tool for teaching programming to children, asserts that in two weeks of instruction, children who were taught LOGO could program circles around children of the same age being taught BASIC for comparison.

But LOGO has so far been a washout for political reasons.

Picture the situation if you will. Some extremely bright and visibly eccentric people, who have very little respect for

Introducing
the PUP-1 Micro
from SEALS....



Seals is proud to announce the Peripheral Universal Processor (PUP-1). The PUP-1 is a truly continuous-duty microcomputer, designed to deliver reliability, performance and maintenance-free operation. It is a complete micro processor with dual built-in floppy disk units and 32K memory standard. The PUP-1 is ideal for educational, business and OEM applications.

The PUP-1 is designed with the same attention to detail that has given all Seals products an excellent reputation within the industry.

If your local computer store does not have information on the PUP-1 and other Seals microcomputer products, contact the factory direct for further information, including a list of retail dealers in your area. Send inquiries to Seals Electronics, Inc., 10728 Dutchtown Road, Concord, TN. 37922, or call 615/966-8771.

SEALS ELECTRONICS, INC.

The Seal of Performance

computer programming as it is ordinarily done, have been saying that computer programming should be taught to very young children in a way that most computer programmers don't understand. They have asserted that this scheme will make the children better programmers than the professionals; and they have sought funds to carry on this teaching in schools where nobody knows what a computer is at all.

Such is the computer field.

Another Lambda language which may become important is TRAC language, invented by Calvin N. Mooers, the same man who brought you the phrase "information retrieval." (Mooers may sue me if I neglect to mention that TRAC is the trade mark and service mark of Rockford Research, Inc., 140½ Mt. Auburn St., Cambridge, MA 02138. He does make things difficult for those who try to use it without his permission.)

Once FORTRAN was a milestone; now it's a millstone.

TRAC language will run on a much smaller computer—one authorized version of TRAC language runs in only 8K spaces of the main hobby computer. TRAC language is like LISP in that it uses many parentheses. Computer people who have been turned off to LISP—and that seems to be a lot of people—see the parentheses in TRAC and say, "Forget it." People who only know BASIC often have the same reaction.

But TRAC has certain special qualifications which ideally suit it for the very small computers that are now becoming so very widespread. It does not need large amounts of memory, and it has important features for highly interactive systems. The ability to control user input, so that if a user types the letter *F*, he instantly sees, say, a picture of fish instead of the letter *F*, is an extremely important feature for user-level systems of the future.

The last Lambda language we will mention here is probably the most exciting. It is called SMALLTALK and was devised by Alan Kay and his associates at the Xerox Palo Alto Research Center. It's written up with neat pictures in the September 1977 issue of the *Scientific American*, pages 231-244.

This language was created around Kay's notion of a personal computer, which he calls a "Dynabook." (Apparently the term *Dynabook* simply means a computer that you can program with the SMALLTALK language.) But Kay and his associates have proceeded on the correct assumption that it would be possible within a few years to build a computer the size of a book that will run on batteries, have an elaborate graphics screen, and sell for $400.

People were alarmed by LISP because it contained hundreds of parentheses.

This prediction, which seemed outrageous to some people only a few years ago, now seems firmly possible for the year 1980. Whether the management of Xerox, deeply entrenched in a paper-oriented way of thinking, will understand this development and bring it to market, remains to be seen. SMALLTALK, anyway, is a Lambda language with numerous exciting features. The parentheses are few, not the tangle of LISP. Instead, some commands of the language consist of smiling faces and pointing hands, among the other symbols and phrases.

Secondly, the language is set up for the use of a finely detailed computer screen, of some half-million dots, on which the programmer may typewrite in numerous typefaces. SMALLTALK may produce dazzling animations on the screen, interacting with the user. (In another amazing form of interaction, Kay hooks SMALLTALK up to an organ keyboard coming out of loudspeakers through the computer. At the same time, the SMALLTALK program shows the notes on the screen transcribed from his pressings of the keys.)

SMALLTALK programs are sectioned into a number of parts, called "processes," which are independent entities with a special kind of autonomy. Processes cannot interfere with each other, and thus a program may be debugged, or corrected, by sections.

But numerous copies of a process may exist. SMALLTALK programs, amazingly, are much more "like real life" than most computer programs. For instance, if you write a program to simulate traffic, you have one copy of the "car" process for each car on your highway.

If you've done ordinary programming, you know how odd that seems to most programmers. Yet it has an intuitive simplicity. Thus SMALLTALK may turn out to be both the most powerful computing language and the ideal language for beginners. (Let's hope the Xerox management gets moving on it.)

Other Languages, Especially APL

There are many other languages; some have very specific ranges of purpose, others are "general purpose" but reveal a certain slant and certain special aptitudes. Foremost among these other languages is APL, or "A Programming Language," devised by Kenneth Iverson. Iverson is a fiery and upright figure, with the dignity and self-certitude of a Raymond Massey, or a religious leader.

Iverson claims that his language was always intended as a way of writing things down, especially for mathematicians and scientists, and feigns surprise that it turned out to be "a good way to drive a computer." For Iverson's notation is a powerful and elegant system of expressing mathematical meaning. Having detected, as a young mathematician, that the notations of science and mathematics are really quite chaotic and irregular, he began writing them out in a form which adhered to certain basic rules. Working all this out, he gradually put together a notational system of computer generality.

No attempt will be made to give examples here. (See "APLomania" by Eben Ostby in the August 1977 *ROM*.) But Iverson's language has become one of the most influential forces in the world of scientific computing. APL is a work of art, not unlike a beautiful set of surgical tools, or a set of matched gems.

Iverson's language permits the expression of mathematical concepts from across the whole of science and statistics, thousands of different ideas and functions each resolved to a crisp and concise expression in this new, common form.

The language requires learning new symbols, but a few hours of time spent with an interactive terminal and a good tutor make one able to do astonishing things.

It is interesting to note that APL has come into use almost entirely on a word-of-mouth basis. An ever-growing fraternity of scientists (and, more recently, business users) have discovered its power for a vast assemblage of purposes.

The original APL program was created within IBM, not as a planned product, but as a private project at the initiative of Iverson and his friends. But the language then caught on with IBM, becoming addictive to its users, and became a part of the IBM product line by popular demand from the outside. It is now affecting the rest of IBM's product line, as both scientific and business users work with it more and more.

Lambda languages were first used in obscure research laboratories where delightful odd people worked on AI.

APL is now available for personal computers, especially the 8080. (Prices vary from $10 to $650 for different versions.) One version sells for as little as $10, but the version from Microsoft, a very respectable programming firm, is expected to sell for about $650.

For many purposes, APL is slow and inefficient—for interactive graphics and music especially. But then again, David Steinbrook, a doughty young composer, is using it as a music machine anyway, and maybe he's onto something.

IBM sells a small computer that runs APL. This is one of IBM's best products. However, because of its cost ($5000 to $15,000), we will not consider it here as being within the range of personal computers.

Other Non-Standard Languages

There are fifty or a hundred languages that ought to be mentioned. But you can see there is no room for that here. The different languages embody different ways of thinking, different styles, different purposes. Many are variations of ALGOL. (If you want to immerse yourself in the great range of them, Jean Sammet's monumental book on programming languages is surprisingly readable.)

Suffice it to say that if you get serious about computer programming, you can make computer languages your never-ending study. Or if you go to do research at the Gazerkis Institute of Tough Science (if there is such a place), you will probably become a fan of their language and see no other. ▼

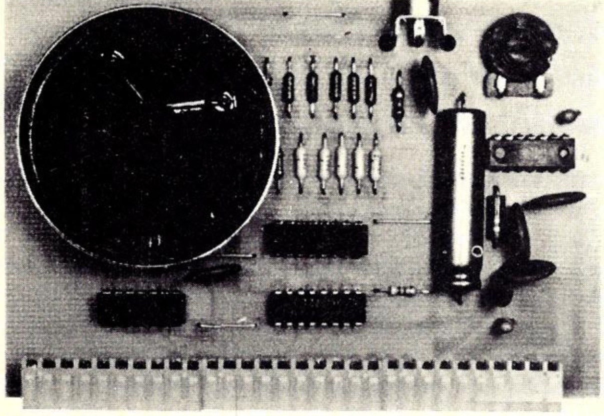
NEWTECH
Model 68

PRODUCES MELODIES, RHYTHMS, SOUND EFFECTS, MORSE CODE, TOUCH-TONE SYNTHESIS, AND MORE!

AS SEEN ON NATIONAL TV!

MUSIC BOARD

- Compatible with Southwest Technical Products Corp. 6800 computer system. Uses one I/O slot.
- Glass epoxy printed circuit board with high quality components.
- 6-bit latching digital-to-analog converter.
- Audio amplifier
- Speaker
- Volume control
- RCA phono jack for connection to external speaker or audio system.
- 60-day parts and labor warranty



Complete user's manual with BASIC program for writing musical scores and 6800 Assembly Language routine to play them.
SWTPC AC-30 compatible cassette with above programs.

$59.95

AVAILABLE THROUGH YOUR LOCAL COMPUTER STORE

ASSEMBLED AND TESTED

NEWTECH COMPUTER SYSTEMS, INC.

131 JORALEMON STREET * BROOKLYN, NEW YORK 11201 * (212) 625-6220

AIQuotient

ROM'S ROBOT REVIEW PART II: QUASAR— FACT, FANCY, OR FRAUD

by A. I. Karshmer

Quasar Industries, Inc., a small New Jersey-based company, claims to have created the first in a series of home robots which can carry out tasks around the house. Their product, which resembles a five-foot-two aerosol can, is said to have a speech-recognition system with a 4800-word vocabulary, a sonar-style navigation system, and the ability to carry out household chores—vacuuming, serving dinner and cocktails, babysitting, answering the doorbell, and greeting visitors. This highly-publicized robot, with the unlikely name of Sam Strugglegear, has enjoyed national publicity in *Newsweek, Parade,* and the *New York Times,* not to mention numerous lesser-known magazines.

Has Quasar Industries actually made a breakthrough in robotics, or is the company perpetrating a gigantic fraud on a microprocessor, gadget-crazed buying public? This month I will present some interesting data regarding the Quasar robot and the claims made about it, so you can come to your own conclusions. Keep in mind, however, that it is very difficult to evaluate any complex system, such as a robot, when the manufacturer refuses to divulge any technical details pertaining to its product.

Recently, an interesting report appeared on the ARPA Net which was entitled "The Carnegie-Mellon University Artificial Intelligence Lab Meets 'The Ultimate Home Appliance.'" The ARPA Net is a huge computer network supported by the Advanced Research Projects Administration of the Department of Defense. The Network has become an exchange medium for researchers in the artificial-intelligence community as well as being a computational system. This report was sent by a well-known AI researcher at Carnegie-Mellon to many of his colleagues around the country. A short preface to the report stated:

In the men's department, among the three-piece suits, was a five-foot-two aerosol can on wheels talking animatedly to the crowd.

> When *Newsweek* called me a few months back to talk about the Quasar robot, I said it had to be a fake, although it might be possible to build an ELIZA-like system that pretended to understand speech.
>
> Yesterday, the robot was on display in Pittsburgh. The following report by Mark Fox shows that it is indeed a fake.

After reading this report, I immediately contacted Mark and obtained his permission to reprint his report in this issue of *ROM.* I think that you will find it as fascinating as I did.

(mss. # 4, 7478 chars)
Mail from CMU-10A rcvd at 19-Dec-77 1112-PST
Date: 19 Dec 1977 1408-EST
Sender: MARK FOX at CMU-10A
Subject: robot letter
From: MARK FOX(C380MF21) at CMU-10A
TO: nilsson at SRI-KL

> The following letter is to be forwarded to Art Karshmer at the University of Massachusetts at Amherst.

The Carnegie-Mellon University Artificial Intelligence Lab
Meets
"The Ultimate Home Appliance"

(Reported by Mark Fox and Brian Reid)

On 24 October 1977, a well-known department store in the heart of Pittsburgh advertised the appearance of a "domestic robot" named Sam Strugglegear. Although this robot is not yet offered for sale, its inventor, Anthony Reichelt of Quasar Industries in New Jersey, claims that its powers include speech recognition with a 4800-word vocabulary, sonar-navigated steering, and the ability to do household chores such as vacuuming, serving drinks, and babysitting. This highly-publicized "robot" has been described in *Newsweek, Parade,* and other national magazines.

Knowing of CMU's pioneering work in Artificial Intelligence, particularly in the field of speech recognition, various friends have called CMU to ask how this robot might be so much better at speech recognition than our talented and dedicated research team.

Rising to the challenge, four courageous members of our department went downtown to investigate. They found a frightening sight: in the men's department, among the three-piece suits, was a five-foot-two image of an aerosol can on wheels, talking animatedly to the crowd. The robot seemed able to converse on any subject, to recognize the physical features of customers, and to move freely (though slowly) in any direction. While the crowd was quite charmed by the talented machine, we were dubious, and moved in to investigate it more closely.

The robot moved on a set of wheels: there were two large drive wheels about ten inches in diameter, and several small stabilizing wheels—a mechanism quite similar to the MIT turtle. It moved about three inches per second, approximately one-tenth the normal walking speed of an adult. We saw both arms rotate at the shoulder along a horizontal axis. Although there was a joint at the elbow, we never saw it move (perhaps this model had no actuator in the elbow).

The hands were like clamshells in design. There was a rod at the wrist that could be used for opening and closing

Illustrated by Cindy Hain

the hands, but on the model we saw, the hands were actually glued shut, so that they could not move even if there were an actuator. The actuators for the arms were electric motors attached to the arms by gears rather than belts. When an arm was blocked while in motion, the motor would stop dead, indicating the presence of some primitive feedback mechanism. One patron asked to see the robot vacuum a carpet, but was brushed off with the reply that its batteries were running low.

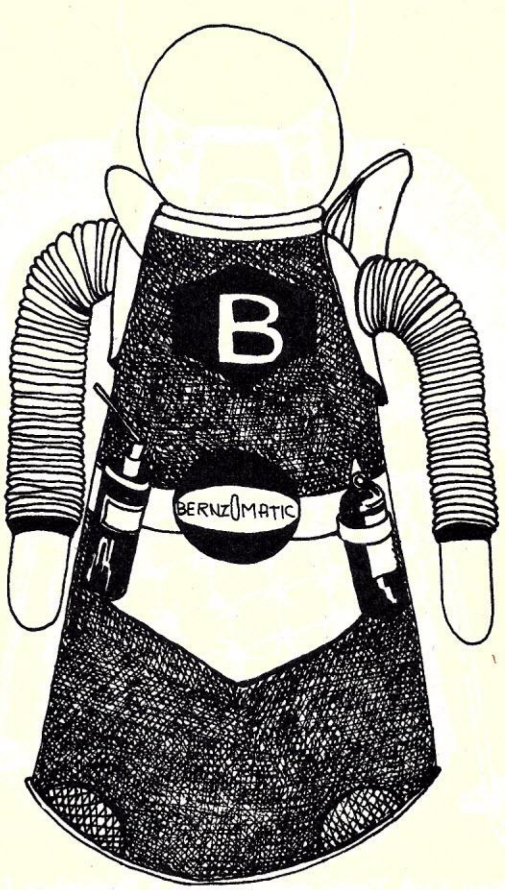
The CMU team next set out to investigate the robot's sensory mechanisms. Pushing and blocking its motion had no effect; the motors kept spinning away. It didn't seem able to tell that an object was blocking its path. Covering the faceplate did not change its behavior at all. Since the robot seemed able to navigate around the room without hitting anything, we found it quite curious that it had no detectable sensory reactions.

Feeling more dubious, we began looking around the room for evidence of remote control. Lo and behold, about ten feet from the robot, standing in the crowd, we found a man in a blue suit with his hand held contemplatively to his mouth like Aristotle contemplating the bust of Homer in the famous Rembrandt painting. After watching for a while, we noticed that whenever the robot was talking, the man in the blue suit could be seen muttering into his hand. Further, seeing that this man had a wire dangling suspiciously from his waist to his shoe, one of the CMU group screwed up his courage and approached this stranger. "Do many people figure out what you are doing?" we asked. "No," he said, "they are usually too busy watching the robot to notice me." "Aha!" we thought to ourselves, "It looks like we're on to something here."

We then asked him what were the robot's speech and vision abilities. To which he replied that the machine can see about ten inches, dimly, and that its speech-understanding ability was about 200 words of unconnected speech in a quiet environment.

We didn't really believe his statement of the robot's abilities, and the light of our discoveries of the robot's poor perceptive skills, we were convinced that there must be yet another remote control handling the motion. Time was running out; they needed to move the machine to a suburban store for an evening demonstration. We returned to CMU feeling unsatisfied.

When we gave our report to the rest of the lab at CMU, a second group of eight immediately set out to the suburban store, determined to find the source of the robot's control. They found a furtive-looking and rather disagreeable person loitering in the back of the room. He was carrying an airline flight bag, with his hand stuck down inside the bag. We asked him his business, and he replied that he was a truck driver. He became extremely agitated when we asked him what was in the bag, asking if we were police. We dispatched a person to watch him, in an attempt to find correlations between the movements of his hand and the movements of the robot, whereupon he got very excited and called for store officials to come get us away from him. We never did get to see in the bag. However, we did see the man with the microphone say to a store official, "Tell him we want to take it for a walk." The store official then wandered over to the "bag man" and whispered something to him.

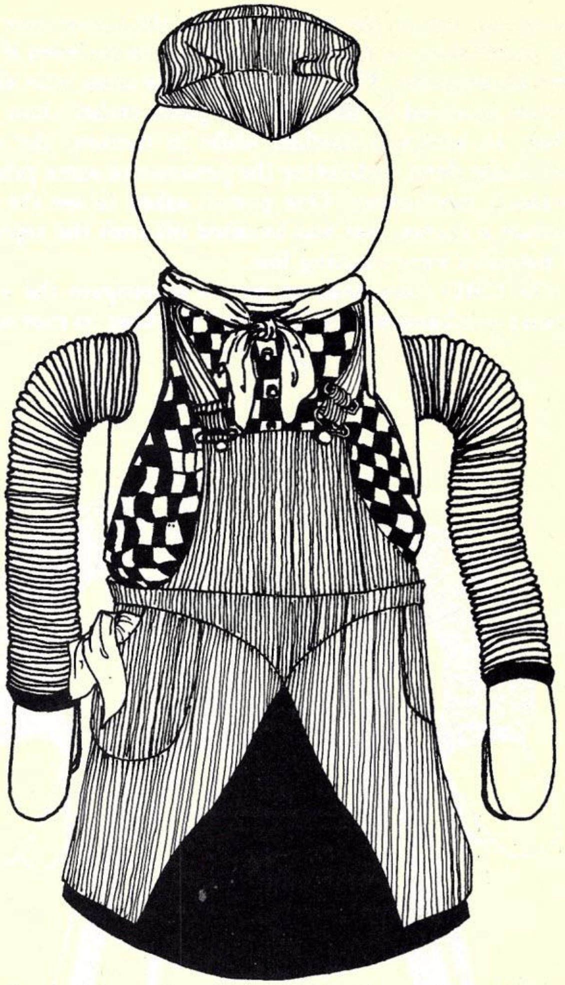


It would be tempting to call this robot a fake, but it is not. It is a fake robot, but a reasonably good parlor trick, more in the domain of magicians than of computer scientists. However, one is reminded of how much better were the parlor tricks of olden days. For example, the chess-playing robot built by Baron Wolfgang von Kempeler in 1769. Spectators were given a view of the inside of the robot, satisfying themselves that it could not possibly contain a person. The robot would then trounce them at chess, all the while rolling its eyes and nodding its head. The workings of this famous "Turk" were not revealed until 1848, more than seventy years later, when it was bought by the Philadelphia Chess Club and disassembled. Thousands of people, including Napolean and Edgar Allan Poe, tried unsuccessfully to figure out how it worked; very rarely was it even beaten.

Kempeler's description of his own robot, circa 1771, is probably the best summary of Sam Strugglegear: "A mere bagatelle, not without merit in point of mechanism, but whose effects appear marvelously only from the boldness of conception and the clever choice of methods adopted for promoting the illusion."

A *New York Times* article dated 5 December 1977, by N. R. Kleinfield, entitled "Developer Calls It a Robot; Scoffers Say Screws Are Loose," deals with the Quasar robot and how it is viewed by the AI community.

> "I don't think there's a ghost of a chance that a mad scientist could come up with this, no less Quasar," says Marvin Minsky, a science professor at the Massachusetts Institute of Technology who has been working on getting machines to see since 1963. "They can't realize all these claims, there's no doubt about that. If you spent a lot of money, you might get something in 1990 that could do some of the things they claim."



John McCarthy, the Director of the Stanford University Artificial Intelligence Laboratory, was quoted as saying:

> I think it's a preposterous fraud. The state of the art is nowhere near this—not in voice recognition, vision recognition, or motion.

Finally, Kleinfield reports the following responses by Anthony Reichelt, the president and founder of Quasar Industries.

> May God help me if it isn't [real]. What happens if I sell a thousand robots and they don't work? I'll be in jail. I'll be sued to high heaven. I tell you I've got the finished robot to prove our claims. I've got the goods. They put Marconi down, they put Newton down. Every major scientist has been put down. To everyone who has put us down, I say, "Just wait and see, buddy."

Well, Mr. Reichelt, I really hope you do "have the goods," and if, indeed, you do have them, I would like to extend an invitation to you to supply us with more technical details. I will be more than happy to print any reasonable response that you might have to your critics. I think the time is long overdue for you to clear the air of suspicion and doubts regarding your "amazing" robot. ▼

Let's Get Personal in Anaheim

June 6-8, 1978

A rewarding personal experience is in store for you June 6-8 at the NCC '78 Personal Computing Festival… the most comprehensive personal computing event ever held. The Festival, a separate feature of the National Computer Conference, will include approximately 30 program sessions, commercial exhibits of consumer computing products and services, plus a contest and exhibit of microprocessor systems and applications. All Festival activities will take place in the Disneyland Hotel Complex, just a few minutes from the Anaheim Convention Center, site of this year's NCC.

Plan now to attend the big, new NCC '78 Personal Computing Festival. The program will include special paper, panel, and tutorial sessions on such topics as speech synthesis and recognition, computerized music systems, hardware and software design, computer graphics, and small business systems. All papers will be published in a softbound volume, *Festival Digest '78,* which will be available during NCC.

Festival exhibits will provide an extensive display of commercial offerings by organizations serving the personal computing field. More than 100 companies, occupying over 175 booths, will display systems, components, terminals, software, kits, disc and tape cassettes, relevant publications, and related hobby items.

Rounding out the Festival will be a contest featuring microprocessor systems, devices, and applications ranging from home-brew DOS and graphics terminals to educational applications and computer games. Prizes will be awarded for the best exhibits.

Don't miss the year's most exciting personal computing event. For more information, return the coupon or call AFIPS at 201/391-9810.

☐ Please keep me up-to-date on Festival plans and activities.

☐ My company is interested in exhibiting at the Festival.

☐ Please send me information on the special NCC Travel Service.

Name ____________________

Company ____________ Division ________

Street ____________________

City ________ State ________ Zip ______

NCC '78
Personal Computing
Festival

ROM

c/o AFIPS, 210 Summit Avenue
Montvale, N.J. 07645
telephone: 201/391-9810

The Human Factor
ALL KEYED UP

by Andrew Singer

Sooner or later, anyone who commits monthly journalism finds that there are odds and ends lying around. As a regular writer, I am always on the lookout for new sources of inspiration and fresh material. Unfortunately, a lot of the things that turn up are just scraps—too small to form the basis for an entire column or article. It seems a shame to discriminate against these leftovers just because they're small. So this month, in this column, it's smattering time.

Not everyone is aware of it, but the fact is that the standard-typewriter-keyboard layout is less than optimal. Actually, key arrangements have been designed that enable a typist trained on them to achieve significantly greater typing speed than with a regular keyboard layout. Because the typewriter keyboard has been standardized for such a long time, I had always assumed that its defects were due to a lack of human-factors knowledge on the part of "primitive" designers.

So I was quite surprised when, in a casual conversation, a colleague of mine pointed out that the typewriter keyboard had been very carefully engineered by those early designers to slow the typist down, and was, in fact, a pretty sophisticated piece of work. And why, you might ask, would anyone want to slow a typist down? Well, I have heard two possible explanations. The first is that the mechanical linkages in early typewriters could not accommodate great typing speed. The second explanation is less logical, but more likely: apparently no one wanted to introduce a machine which would go so fast as to embarass stenographers. And there I was, all that time, grumbling at the lack of sophistication of those "primitive" designers.

I should have known better. Among those designers are numbered men and women like Frank and Lillian Gilbreth, early pioneers in the struggle to make technology pleasant for people. No one, as concerned with wasted time and motion as the Gilbreths were, could have overlooked such a crucial item as the typewriter keyboard. Indeed, we have it on good authority that Frank Gilbreth did not overlook the typewriter keyboard. In *Cheaper by the Dozen,* his son, Frank, Jr., vividly describes the day that Papa brought home one of the first Remington Noiseless typewriters. Curiously, the typewriter was entirely white. As the eight or nine Gilbreth children (I can't remember how many Gilbreth children were actually born at that point) were to discover, the white typewriter was no eccentricity. Frank Gilbreth had an idea about training people to touch-type, and he intended to try it out on all of his children. After all, as he pointed out, if he could teach a young child to type, he could teach anybody to type.

His system was simple. What he did was color code the key tops and then paint the corresponding fingers of the trainee's hands with the corresponding color. Gilbreth was well aware of the value of motivation, and so he provided his experimental subjects with a strong stimulus to perform well. (I seem to remember something about a $1.50 reward when the child reached a certain speed, but, again, my memory fails me on the exact nature of the incentive.) In any event, the system was apparently quite successful, except for the fact that the dyes didn't come off very readily and the children went about for weeks with hands that looked oddly diseased. Of course, the typewriter was white, which made the colors stand out. My typewriter is blue, has gray keys, and I am afraid I still can't touch-type, but I can hunt-and-peck at a high speed—the result of many hours spent at the tender mercies of one terminal or another.

Actually, being able to touch-type is a distinct drawback on many computer terminals because the touch of the keys is so bad. Then, touch-typing generally involves a reasonable flow of work, whereas typing at a terminal is staccato with many hiccoughs. A human-factors psychologist that I know, who has done a lot of studies with terminals and keyboards, tells me that the "feel" of the IBM Selectric keyboard is absolutely optimal. I wonder how many typists would agree with him.

I find it interesting that companies like Cannon and Burroughs can survive by marketing large-key versions of pocket calculators at substantially higher prices. Sometimes the performance of these more expensive calculators, in terms of their capabilities, is rather poor in comparison to the less-expensive machines. The best explanation I have heard for the survival of these companies is that in a production environment, key-feel is more important than cost or capabilities.

Certainly the keyboards of inexpensive calculators vary widely in touch. My own preference is for the Hewlett-Packard type of key. I least like touch-sensitive keys. Recently, however, I used a device whose touch-sensitive keys were backed up by a beeper. The auditory feedback of key depression provided by the beeper eliminated a lot of my objections to the lack of feel in the touch-sensitive keys.

This seemingly perverse obsession of mine with keys and keyboards is not without reason. Increasingly, people are discovering that special-purpose keyboards are an effective way to enable the ordinary person to get at the power of a computer. Of course, special-purpose keyboards have been around for a long time. Consider the piano. A piano is a special-purpose, keyboard-controlled, mechanical, music synthesizer. Modern calculators, particularly the more sophisticated ones, are another example. But designers are branching out. For instance, there is a new, very sophisticated, industrial-control module built around a microprocessor that is entirely programmed by means of a special-purpose keyboard in the side of the module. And the most elaborate special-purpose keyboard is the one designed to work with ECD's MicroMind word-processing system. Although I have not actually seen this device, I have it on good authority that it is beautifully designed and puts virtually all the editing commands on individual keys.

The power of special-purpose keyboards does not result just from decreasing the number of key strokes necessary to do something. It derives more from the fact that a single key becomes an effective symbol for a single concept. An entire keyboard becomes a conceptual menu, or index, for the device it controls.

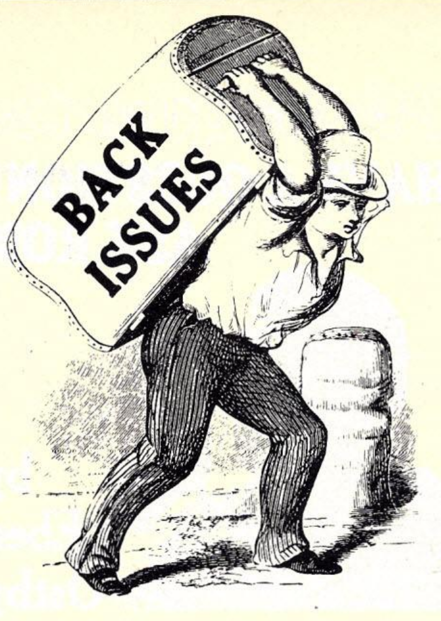
At Bell Labs, a gentleman named Kenneth Knowlton has carried the special-purpose keyboard a step further. Using a keyboard with blank keytops, a CRT display, and a half-silvered mirror, he has devised a dynamically programmable keyboard. The user sees, superimposed upon the keys, whatever symbols or legends the CRT projects. This makes it possible to create multiple special-purpose keyboards under computer control. An advantage of the scheme is that the operator's hands do not obscure the legends. Knowlton has worked up some fascinating examples of how the keyboard can be used. (For details, see the March, 1977 issue of *The Bell System Technical Journal.*) While the physical arrangement is a bit awkward, the idea is lovely.

I am still waiting for some manufacturer to introduce keys with embedded five- or six-character alphanumeric LED displays. Maybe next year.

Of course, if you are looking for the ultimate in special-purpose keyboards, there is really nothing that beats the touch display. Briefly, a touch display consists of a CRT or a plasma-display screen which has, built in front of it, some sort of sensing device that can determine the position and presence of a finger. Two ways of doing this are currently in use. The PLATO terminal, which uses a plasma screen, has rows of photocells along the sides and bottom of its screen. Opposite each photocell is a matching infrared light source. When you put a finger anywhere on the screen, it breaks these light paths, and the terminal can identify where the finger is. By correlating finger position with the position of messages displayed on the screen, it is possible to deduce what the touch means. This is put to use quite nicely in PLATO tutorials for young children: point to the frog you want to put in a bag; point to the bag you want to put it in; presto! the frog is in the bag. Another type of touch-sensor uses a grid of fine wires or transparent metal electrodes to electrically sense the presence of a finger. A very powerful medical-record-keeping system has been developed around this type of touch display at the University of Vermont. Touching particular items on the screen enables you to expand the data about that item, and touching an item in the expanded screen enables you to expand further, and so on. People who have used this system have described the experience to be like using your finger to direct you through space. Touch and you are in another room. Touch and you have gone through a wall.

Touch displays are really something quite beyond keyboards. Because they provide all of the advantages of single-key control and are completely flexible as well, I believe that they will replace keyboards in the future. "Let your fingers do the talking," may well be the motto of future computer users.

On looking over this column, it appears to me that my "scraps" fit together a bit too well to be called a smattering. That's what happens when I'm really interested in a topic. I get all keyed up. ▼



Look what you've missed!

Complete your collection
NOW!
Back issues $3.00 postpaid from
ROM Publications Corp.
Route 97
Hampton, CT 06247

Also still available—July, August, September, October, and November back issues.

February '78
Computers Come to Chelsea
The Mailing List Program
Up and Running at the Elections
Artificial Intelligence
Assemblers
Flowgrams—A New Programming Tool
Western Easters and When They Come
The Wheelbarrow Thief (A short story)
Even More BASICally
Translate!

January '78
Synthetic Skin for Your Robot and How to Make It
The Code That Can't Be Cracked
TLC: The Visual Programming Language
A Beginner's Guide to Computer Graphics: or Raster Scan Can
Video-Sketch: Your Computerized Drawing Board Program
Home Computers: A Look at What's Coming
The Computer and Natural Language
Microcomputers Help the Deaf-Blind
First Timer's Guide to Circuit Board Etching
Call Exit (A short story)

December '77
Computers Challenge America's Cup
A Beginner's Guide to Peripherals
Input/Output Devices Your Mother Never Told You About
Computer Country: An Electronic Jungle Gym for Kids
The Best Slot Machine Game Ever
The Micro Diet: Better Health through Electronics
Come Closer and We Won't Even Have To Talk
The Kit and I, Part Four: Testing, Testing
Putting Two and Two Together: Part 0010
Computer Models in Psychology
Fair Play (A short story)
Micro, Micro on the Wall,
How Will I Look When I Am Tall

PROMqueries

HAVE A QUESTION? ASK ROM

by Eben Ostby

Dear ROM,

I have a very limited computer system—an 8080-based machine with 8K bytes of memory, a slow teletype, and two cassette recorders. Right now, I can't afford a floppy, although it would be really nice to have. I am trying to set up a simple accounting system, with a small general ledger, a journal, and programs for updating these and producing reports. My problem is that, without a floppy disk, I can't see how to update ledger accounts at will. Can you think of a way around this problem?

Lauren Olney
New York, New York

Dear Lauren,

I take it that you are thinking of having the journal tape on one cassette recorder and the ledger tape on the other recorder. Well, I don't know how many accounts you plan on having in your system or how many transactions you'd like to handle in a day, but you could try storing each day's transactions in the memory of your computer until you're done for the day—that is, until you've entered as many as you want to. Then you could run an update on your ledger tape. What you'd do is this: you'd have two identical tapes with ledger data on them. Each day, you'd take yesterday's tape and today's transactions, update the tape, and write the result onto the day-before-yesterday's tape. After this was completed (it would take some time, since you'd have to copy the entire tape), you could do the same sort of thing with your journal tape: read in the journal tape and write the data out onto the older journal tape, adding today's entries to the end of the tape.

This is known as a "backing-up" system. It's pretty slow, but it has the advantage that you've always got a copy of yesterday's tape around, so if things go awry, you can just repeat a daily update. It's a very popular technique. There is one major limitation, though: you have to store all your transactions in main memory, so you're limited to the amount of space left over when your programs are loaded. If you can scrape up 1000 bytes for storage of the data, you should be able to fit fifty transactions in memory at a time—limited, but not impossibly small.

Good luck!

ROM

Dear ROM,

I've seen lots of articles lately about computers in the kitchen. Is there any connection between the "hash codes" used in computers and the kind Mother used to make? (I won't even mention the kind we used to smoke.)

Barry LaFarge
Orange, New Jersey

Dear Barry,

In the kitchen, hash is what you do with the leftovers. With computers, it's a way of dealing with the stuff you want to keep. If you have a bunch of data and you want to find any particular piece in a hurry, there are a number of ways you can organize things inside the computer to make finding it quick and easy. One way might be just to search through the list until you find what you're looking for. Another way might be to assign each datum to a specific place in a table, based on some identifier, or key. For instance, if you had people's names and social security numbers, and a table with space for a hundred people in your computer, you might take the last two digits of the social security number and use them as an "index" to the table. If a person came along with social security number 040-40-8946, you'd put his name and number in slot 46 in the table. Of course, you'd run into a problem when someone else came along with number 911-25-7446, but there are ways around that problem.

This sort of method is what's known as a "hashing" method. There are many different ways of accomplishing the same thing on a computer; hashing has the advantage of being simple and very fast.

ROM

Dear ROM,

I have read a little bit about algorithms, both in *ROM* and in some of the other magazines. In my reading, I have seen references to a "hidden line algorithm." What is this?

Liz Worth
Oakland, California

Dear Liz,

What you've read about is the hidden-line removal problem: how to remove from a drawing lines that should be hidden by other parts of the drawing.

Suppose, for instance, that you have a TV screen attached to your computer. (Okay, first assume that you have a computer.) Also assume that you know the coordinates of the corners of some object you wish to draw on the TV screen. For instance, you might wish to draw a cube with the corners located at (0,0,0), (0,0,1), (0,1,1), (0,1,0), (1,0,0), (1,0,1), (1,1,1), (1,1,0). It's a fairly simple matter to convert these numbers into some sort of two-dimensional coordinates for plotting on a TV screen or whatever. It might look like figure 1 when it's displayed.

But if you want total realism, you have to get the computer to remove those parts of the picture that you wouldn't normally be able to see: the lines in back of other lines. When you've removed those hidden lines, the picture might look like figure 2. Or perhaps figure 3. Or maybe figure 4. Or, worse yet, figure 1! So you've got to tell the computer which parts of the figure exist and which are just empty space.

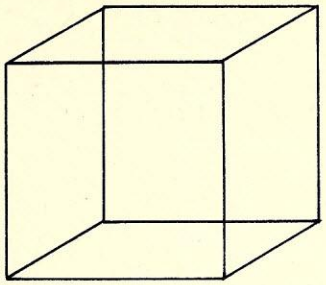


Figure 1

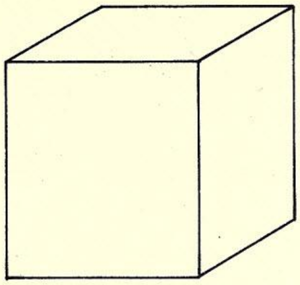


Figure 2

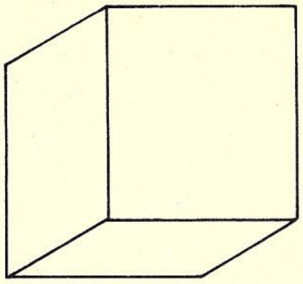


Figure 3

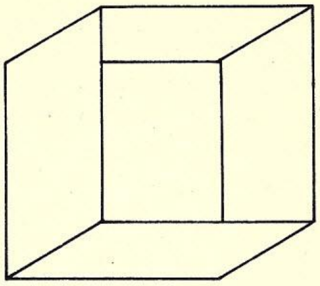


Figure 4

Get the idea?

From this data, it's still difficult to determine—numerically—which parts of the cube are visible and which are hidden. But for an irregularly shaped object, like a building or a car or a person, it's pretty hard to describe what the thing looks like, and even harder to "not draw" the parts that should be invisible. There are a few algorithms—programs—that have been devised to do this, but they are complex and expensive to run. Because they pose some interesting problems, and also because articles about them are invariably illustrated with lots of pretty pictures (that the programs helped draw), these hidden-line algorithms are considered classics among algorithms.

ROM

Dear ROM,

I understand the IBM 360s and 370s have hardware for decimal arithmetic. How is this handled? I thought computers only worked in binary, because numbers were represented by "on-off" pulses of electricity.

Michael Clarke
Butte, Montana

Dear Michael,

You're right—computers do store all numbers in binary. But most binary integers are stored as one long binary number, while decimal numbers on the 360 are stored by digit: each decimal digit is stored as a little binary number. It takes only four bits to store a single decimal digit. There are slight differences in the rules for adding decimal and binary numbers, because you have to treat each group of four bits as a single digit in decimal, while you can add two entire integers directly in binary. But mostly, it's only a slight difference in storage concept that distinguishes binary from decimal.

ROM

Dear ROM,

I haven't got a computer yet, but I'm studying and working towards the day when I'll have my Shangri-La. But one thing bothers me. I hear about have-no-fear-of-crashing-your-system and I can't find any information on it. Please tell me exactly what happens when a system crashes. And is there any damage to the CPU or other components? Thank you very much.

Dave Littell
Jackson, Mississippi

Dear Dave,

Have no fear—a system crash isn't usually like a head-on collision. Actually, a "crash" is anything that makes your computer system, including whatever program happens to be running on it at the time, stop running right. A little software error may cause the computer to stop—that's a crash. Or a chip may get so hot that it melts—that, too, is a crash. But most system crashes are caused by simple program bugs, things like incorrectly stored numbers, or instructions that branch to the wrong place, or loops that run one too many times. Probably the worst thing that could happen when a crash like this occurs is that you might lose some data you typed in. Or perhaps you might lose a morning's work. It can be frustrating, but it's rarely earth-shaking.

The other kind of error, a hardware failure, is much more serious. If a chip burns up, you'll have to have it fixed before your system will run again. If the motor in your floppy-disk drive grinds to a halt, it will have to be replaced. But if you consider how infrequently electronic components break, you'll see that it's a rare occasion for a hardware failure to occur. The worst that usually happens is that the heads get dirty on the disk drive.

In short, you shouldn't worry. Life tends to go on unhampered by system crashes.

ROM

Dear ROM,

I am interested in any information you might give me regarding the application of computer systems to sailboat racing. I am interested in knowing if there are any programs available to optimize the sailing angles and sail selection, as well as any games that simulate a race and related tactics.

John C. MacLaurin
Beverly Hills, California

Dear John,

It isn't terribly common to find full-scale computer systems aboard sailboats, so there is hardly a wealth of, say, BASIC software for racing. However, programmable calculators are turning up everywhere, including aboard ship. Texas Instruments, for instance, has a "solid-state software" module for its higher-priced programmables which includes functions for coastal and celestial navigation, ocean sailing, and tactics. There is also software available for Hewlett-Packard's programmable calculators to assist in racing and navigation.

Computers can and have been used in very sophisticated ways to aid the sailboat racer—take, for instance, the PDP-11 aboard the Cup Defenders written up in ROM's December 1977 issue. ROM plans to run more articles about such applications in the future.

ROM

"DADDY, IS IT THE PET?"

Michael (nine years old) adding a LEM to a Lunar Lander program

Six-year-old Alex at the controls

by Richard Rosner

It finally came. After three and a half months of "Daddy, is it THE PET? Is it THE PET?" (emphasis theirs, in stereo) every time a package arrived, it finally came.

Upon opening up the box and placing the PET in its place of honor on the dining room table, two things became obvious. First, it is a beautiful machine. We all fell in love with it. Second there is a painful lack of documentation. Only a brief brochure listing the commands with some simple examples was included. A note explained that the full booklet will be ready soon.

My immediate inclination, and burning desire, was to plug in the PET, turn it on, and play. However, it was freezing. Literally. It had come three-thousand miles from sunny California and had arrived at 9:15 P.M. in Connecticut to be greeted by snow and twenty-five-degree weather. I was afraid that if I powered it up, the heating effect, especially on the CRT section, would blow something up. Instead, I had to content myself with reading the manual and peeking inside.

According to the manual, the PET has the commands, statements, functions, and operators which are listed in figure 1.

The top of the PET, including the screen, keyboard, and cassette recorder, swings up on a piano hinge and is held in the open position by a supporting rod. The entire computer (except the power supply and video circuitry) is on one printed-circuit board. All the larger integrated circuits, including the ROMs, CPU, and peripheral chips, are in sockets. This makes it very easy to replace defective components and to change the firmware as more advanced BASICs, FORTRAN, and other languages become available.

If you open your PET, be careful. The cassette-recorder cable is short and, if not carefully disconnected when the top is only partially open, it may yank on the main circuit board and break it.

Morning finally came, the PET was at room temperature, and it was time to plug it in and turn it on. A few seconds after the power switch was thrown, the screen lit up with

```
***COMMODORE BASIC***
7167 BYTES FREE
READY.
```

The rest of the day was spent becoming familiar with the machine—its commands and functions, and its program and data file storage capability on the cassette tape unit.

After the first day, time on the PET started to become difficult to get. My

ROMtutorial ROMtutorial

CRT: Cathode ray tube. An electronic vacuum tube containing a phosphor-coated screen on which information may be displayed by the use of controlled beams of electrons; the electrons strike the screen coating, causing it to emit light. Television sets use them.

Cassette recorder storage: Typically, a personal computer has memory, or storage, for a fairly limited number of data items in its fastest random-access memory (RAM). Frequently, memory space is needed for a large amount of data. A good place to put the data is often a simple cassette tape, since it holds a lot and is cheap. There are a variety of methods for adapting normal cassette recorders to work with computers, but most of them convert the on-off signals of the computer into tones that can be recorded easily.

Printed circuit (PC) board: A thin sheet of insulating material to which a copper foil has been bonded; by a process similar to photolithography, the copper foil is selectively etched away so as to leave thin strips of foil, called traces. Electronic components are mounted on a printed circuit board by inserting their leads (the wires that emerge from the component) into holes drilled through the board. Each hole is positioned so as to pass through a particular trace, and the leads are soldered to the traces. The traces thus serve as wires for interconnecting various components on a printed circuit board.

Photographs courtesy of the author

ROMtutorial ROMtutorial

Integrated circuit (IC): A transistor (not a radio) is a tiny, layered speck of specially processed silicon. Integrated circuits contain hundreds or thousands of transistors formed in the same block of silicon, along with the necessary interconnections. The block is still quite small, so it's called a chip. Often it is packaged in an inch-long block of plastic with metal connecting pins coming out from it, and the whole package is called the chip.

Read-only memory (ROM): Memory, or storage, whose contents are fixed at the time the information is entered. It's like a telephone directory the computer can read, but not scribble in. ROM is used to hold instructions (programs) for the computer.

Central processing unit (CPU): The "thinking" portion of a computer. It decides where to move information, what to do with it once it's there, and where it should look for its next instructions.

Keyboard: A device like a typewriter keyboard that enables a person to key information into a computer. The information, in sequences of characters (letters, digits, and special symbols), is called text strings, or simply text.

wife wanted to try some of the problems she had from school—particularly, a compound-interest problem and a depreciation-rate program. My nine-year-old son, Mike, wanted to play games, so he copied *Lunar Lander* and *Chase* from one of the BASIC games books. After that, he spent some time, spread over several evenings, learning BASIC from the book, *My Computer Likes Me When I Speak In BASIC,* by Bob Albrecht. (This is an excellent book for beginners, and takes the reader step-by-step through BASIC's simpler commands.)

Figure 1
PET BASIC

COMMANDS/STATEMENTS

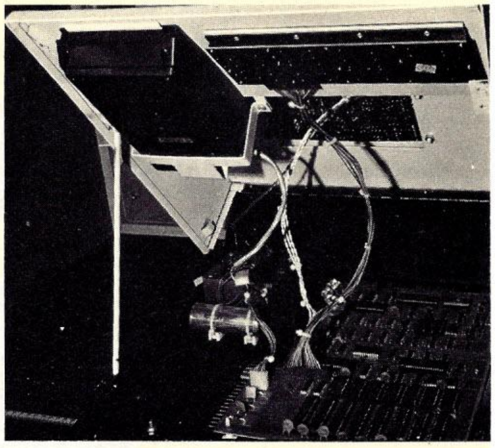
CLOSE	*PEEK*
CLR	*POKE*
CMD	*POS*
CONT	*PRINT (abbr.?)*
DATA	*READ*
DIM	*REM*
END	*RESTORE*
FOR...NEXT...STEP	*RETURN*
FRE	*RUN*
GET	*SAVE*
GOSUB	*SPC*
GOTO	*STOP*
IF...GOTO	*SYS*
IF...THEN	*TAB*
INPUT	*TI$*
LIST	*USR*
LOAD	*VERIFY*
NEW	*WAIT*
OPEN	
ON...GOSUB	
ON...GOTO	

FUNCTIONS

ARITHMETIC	STRING
ABS	*ASC*
ATN	*CHR$*
COS	*LEFT$*
DEF FN	*LEN*
EXP	*MID$*
INT	*RIGHT$*
LOG	*STR$*
RND	*VAL*
SGN	
SIN	
TAN	

OPERATORS

ARITHMETIC		STRING
=	>=	>
+	<=	<
—	=<	=
*	=>	>=
/	<>	<=
↑	>	+
AND		
OR		
NOT		



Inside the PET

My other son, six-year-old Alex, who can just barely read, had to do something too. After he played with the graphics for a while, Mike helped him select a game, and Alex is now, slowly, putting *Awari* into the PET.

Ten days of programming produced these random observations about the features of the PET.

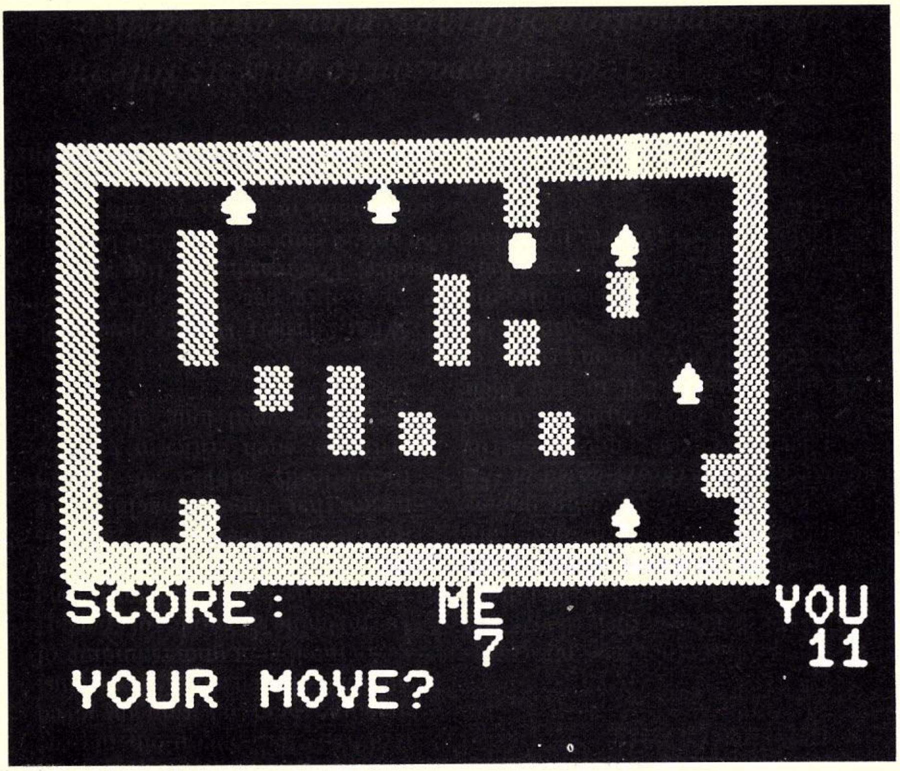
Editing and Graphics Features

One very nice feature of the PET, for the programmer, is its editing capability. If there is an error in a line, it is not necessary to rewrite the entire line. Using the cursor controls (see figure 2), it is possible to insert and delete characters from any line.

The screen configuration is twenty lines with forty characters-per-line. A BASIC line, however, is not limited to just forty characters, but may be as much as eighty characters in length. Each character is made up of dots in an eight-by-eight grid. In addition to the usual alphanumeric symbols and special characters, there are sixty-two graphics characters and a key for reversing the character and the background. With these features, the possibility for using graphics in the home and small businesses becomes enormous.

Diagnostics

One of the first observed features of the PET is its error messages. They are



Chase—five robots in pursuit

printed in English and are not just error-message numbers. These messages say things like:

SYNTAX ERROR IN 10

OUT OF MEMORY ERROR

DEVICE NOT PRESENT

ERROR

BAD SUBSCRIPT ERROR

Digits Displayed

The PET displays nine significant digits. These numbers are represented as integer, floating point, or in scientific notation (exponential form) as the size of the numbers change.

Variable Names

Variable names are stored as two alphanumeric characters. The first being alphabetic and the second being either alphabetic or numeric. While the variable names are stored as two characters, the variable names in the program may be any length (there are some exceptions). For instance,

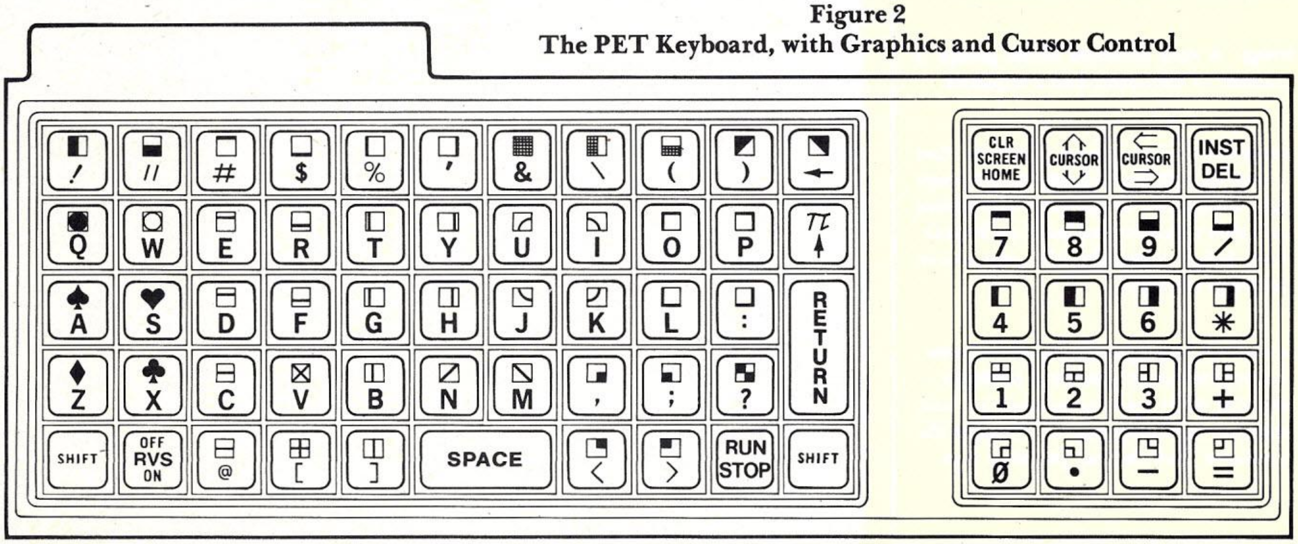
Acceptable variable names	Stored as
ACT	AC
XAXIS	XA
YAXIS	YA
A1234	A1
DIPHERENCE	DI
PRINTER	PR
THIS IS A TEST	TH
Y LIMIT	YL

The exception to this rule is that the variable name may not contain a sequence of letters that spell a BASIC command word. For instance,

Not an acceptable variable name	Because it contains
DIFFERENCE	IF
POSITION	POS and ON
PRINTER	PRINT

Consider the program for compound interest that my wife was working on. This program computes the future value of some principle, given the time for compounding in years, the interest rate, and the number of compounding periods per year. The variables could be written as single letters, thus saving program space. In this case, however, *REM* statements would be required to make the program more understandable, thereby taking more program space.

Figure 2
The PET Keyboard, with Graphics and Cursor Control



ROMtutorial ROMtutorial

BASIC: Acronym for Beginner's All-purpose Symbolic Instruction Code. An easy-to-learn, easy-to-use programming language especially adapted for use with mini- and microcomputers, as well as time-sharing systems. It provides anyone using the computer with instantaneous feedback on whether he's doing all right or making a mistake.

FORTRAN: An acronym for Formula Translator. FORTRAN is one of the oldest and most widely used programming languages. Much of its use is in scientific applications, since its strength lies in its handling of numeric quantities.

Byte: A piece of information consisting of eight bits. It has 256 possible values.

Cursor controls: Cursors are pointers or windows used in graphic display devices to single out information from an image being displayed. Cursor controls are keys or joysticks which can move the cursors.

Alphanumeric: Information made up of sequences of characters containing letters (alphabetics) and digits (numerics).

Random-access memory (RAM): Memory like a set of pigeonholes into any of which the computer can put new information or from any of which it can read old information. The computer can choose any pigeonhole (or address) at any time. RAM can store and recall information at high speeds, and information stored in it can be changed at any time.

Array: A matrix or table of numbers. An array may also be a list of numbers.

String: A data structure which groups a number of characters into a sequence.

Interpreter: A computer program that translates each source language statement into a sequence of machine instructions and then executes these machine instructions before translating the next source language statement.

Software: The programs run on a computer. A computer system consists of hardware—the computer and its accessories—and software—the programs which make the hardware work the way you want it to.

Subscripts in Arrays

Subscripts for arrays include 0. For example, the statement

10 AB(5)

reserves six elements for the array *AB* with subscripts *0,1,2,3,4,* and *5.*

Storage Requirements

The *FRE* function returns the number of bytes available to the user for program space and data storage. The commands, statements, and functions are tokenized before they are stored in the program space. This means that each command, statement, or function is stored as only one byte, regardless of the length of the word, as it is typed or printed. For example,

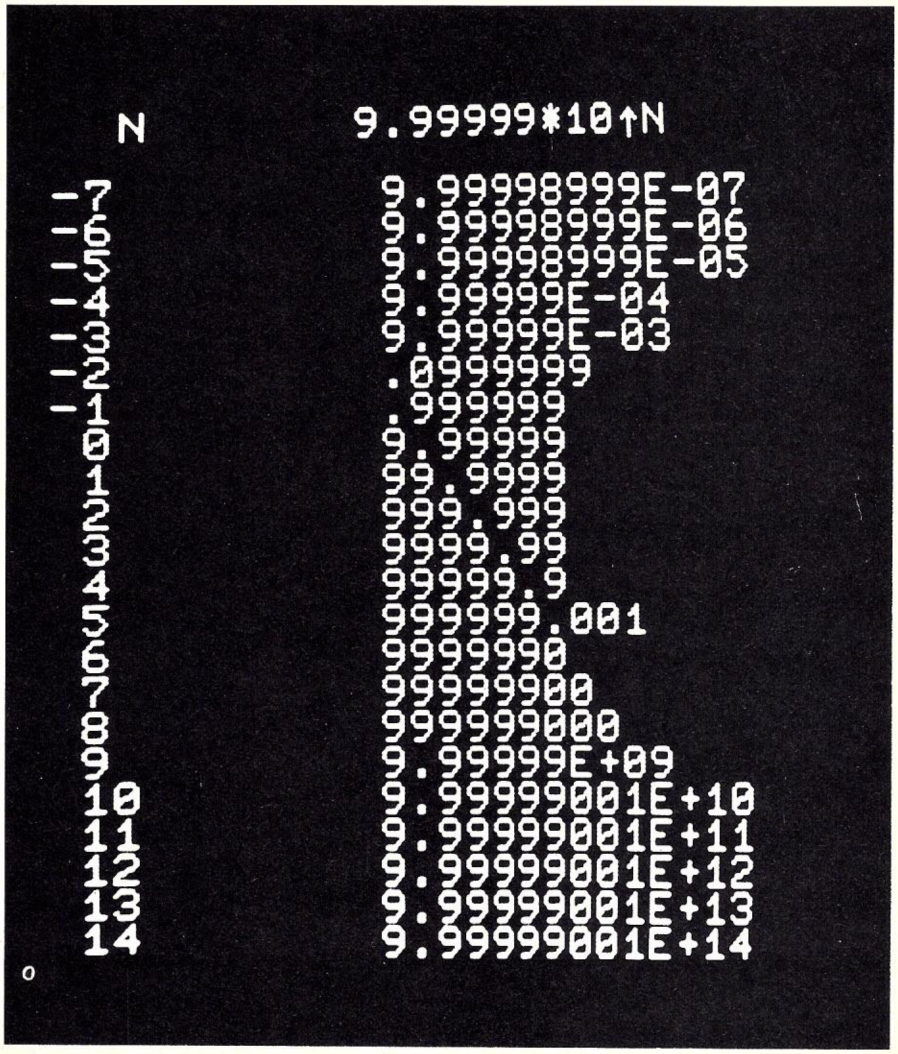
1060 PRINT SIN(A1)

the command *PRINT* and the function *SIN* are each stored as one byte. The entire line requires twelve bytes of the user RAM space. This is broken down as

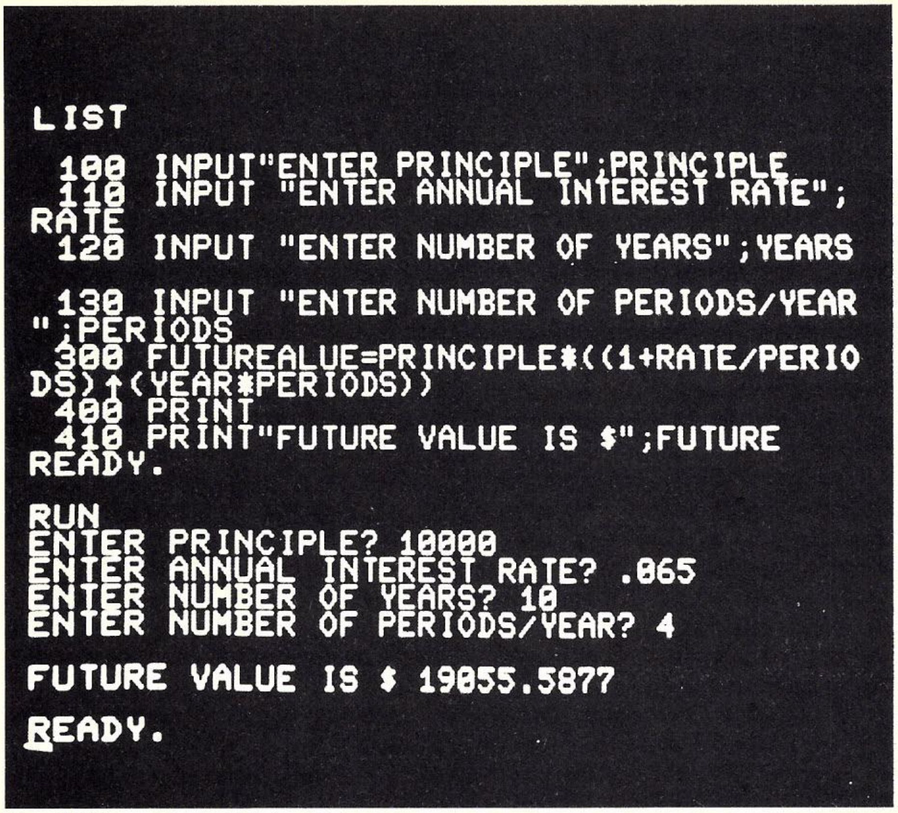
	bytes
Line number and end of line identification	5
PRINT	1
Space between *PRINT* and *SIN*	1
SIN	1
(	1
A1	2
)	1
Total	12

Undimensioned variables:
Each floating-point variable requires 7 bytes.

PET numbers—in integer, floating point, or exponential form



```
   N      9.99999*10↑N
  -7        9.99998999E-07
  -6        9.99998999E-06
  -5        9.99998999E-05
  -4        9.99999E-04
  -3        9.99999E-03
  -2        .0999999
  -1        .999999
   0        9.99999
   1        99.9999
   2        999.999
   3        9999.99
   4        99999.9
   5        999999.001
   6        9999990
   7        99999900
   8        999999000
   9        9.99999E+09
   10       9.9999001E+10
   11       9.9999001E+11
   12       9.9999001E+12
   13       9.9999001E+13
   14       9.9999001E+14
```



```
LIST

 100 INPUT"ENTER PRINCIPLE";PRINCIPLE
 110 INPUT "ENTER ANNUAL INTEREST RATE";
RATE
 120 INPUT "ENTER NUMBER OF YEARS";YEARS

 130 INPUT "ENTER NUMBER OF PERIODS/YEAR
";PERIODS
 300 FUTUREALUE=PRINCIPLE*((1+RATE/PERIO
DS)↑(YEAR*PERIODS))
 400 PRINT
 410 PRINT"FUTURE VALUE IS $";FUTURE
READY.

RUN
ENTER PRINCIPLE? 10000
ENTER ANNUAL INTEREST RATE? .065
ENTER NUMBER OF YEARS? 10
ENTER NUMBER OF PERIODS/YEAR? 4

FUTURE VALUE IS $ 19055.5877

READY.
```

Our PET compound-interest program

Each integer variable requires 7 bytes.

Each string variable requires 7 bytes plus 1 byte for each character in the string.

Dimensioned variables:

Floating-point arrays:

One-dimension arrays require 7 bytes plus 5 bytes for each element.

Two-dimension arrays require 9 bytes plus 5 bytes for each element.

Three-dimension arrays require 11 bytes plus 5 bytes for each element.

Integer arrays:

One-dimension arrays require 7 bytes plus 2 bytes for each element.

Two-dimension arrays require 9 bytes plus 2 bytes for each element.

Three-dimension arrays require 11 bytes plus 2 bytes for each element.

String arrays:

One-dimension arrays require 7 bytes plus 3 bytes for each element plus 1 byte for each character in each string.

Two-dimension arrays require 9 bytes plus 3 bytes for each element plus 1 byte for each character in each string.

Three-dimension arrays require 11 bytes plus 3 bytes for each element plus 1 byte for each character in each string.

A Dimensional Bug

A nasty little bug appeared when I tried to use arrays dimensioned over a certain size. Some experimenting with my PET and another owned by a friend led me to believe that arrays dimensioned over 15-by-16 were wrapping around on themselves. Data put into the top of the array also appeared at the bottom of the array. A call to Gordon French of Commodore verified that this was indeed a bug in the BASIC-interpreter software. It seems that the early PETs cannot dimension a single array correctly when the sum of the total number of elements in the array is 255 or greater.

As with any new device, there are bound to be some bugs in the system. But at some point in the future, according to Mr. French, Commodore will make available to PET owners a new set of ROMs at a reasonable price.

Saving Files on Cassette

Files are saved using the *SAVE* command, followed optionally by a file name which may be up to sixteen characters in length. Files are loaded using the *LOAD* command. If a file is saved twice in succession on the tape, then, after being loaded, a correct loading can be checked by using the *VERIFY* command. This command reads the next, or named, file on the cassette and compares it to the loaded program. If an incorrect load has occurred, an error message is returned. If the command *LOAD PROGRAM* is given, and *PROGRAM* does not exist on the cassette, the entire tape will be searched and each file found will produce the message

FOUND name of file

Time and Timing

The PET has a real-time, twenty-four-hour clock that displays real time

ROMtutorial ROMtutorial

Loops: Segments of a program which are executed a number of times. Usually they consist of a series of instructions followed by a test to see whether the program should execute the preceding instructions, or continue with the instructions that follow.

Hard copy: Information in printed form, e.g., on paper. Soft copy would be information displayed on an impermanent medium, e.g., a TV screen.

I/O: Input/output. The electrical channels through which a computer moves information to and from the outside world.

Table 1

LINE	TIME IN SECONDS
1020 line deleted	1.417
1020 REM	1.733
1020 A=N	2.833
1020 LETA=N	2.867
1020 A= N	2.85
1020 A= N	3.067
1020 A=1	3.267
1020 LETA=1	3.317
1020 A= 1	3.5
1020 A=00000000001	10.333
1020 A=1.0	6.383
1020 A=10	4.23
1020 A =1	3.85
1020 A =9	3.583
1020 A =10	4.65
1020 A=RND(N)	6.9
1020 A=RND(1)	7.35
1020 A=RND(100)	9.267
1020 A=RND(1000)	10.23
1020 A=LOG(N)	25.867
1020 A=SIN(N)	30.067
**1020 A=SIN(1)*	30.68
**1020 A=SIN(1.111111)*	55.63
**1020 A=SIN(2.222222)*	56.283
**1020 A=SIN(B)*	30.28
WHERE B=1	
†*1020 A=EXP(N)*	28.89

*The PET computes trigonometric functions in radians. One radian is 57.2957 . . . degrees. (RAD=DEG*PI/180)

†For the *EXP* function I used a loop limit of 75 and multiplied the result by 1000/75.

as *"hhmmss"* (hours, minutes, seconds). The clock is set to *"000000"* on power up and can be set to any time using the command

TI$="hhmmss"

In addition to the time variable *TI$*, there is the timing variable *TI*. This read-only variable is the time, in sixtieths-of-a-second, since the reset of *TI$* and is calculated from the same clock as *TI$*.

Timing loops become easy and independent of the timing required for software loops. For instance, the two lines

```
1000 TL=TI
1001 IF TI-TL   N*60 GOTO
     1001
```

will produce a delay of N seconds.

One other use of the timing function is determining how long the various PET BASIC functions and commands will take. The program

```
1000 T1=TI
1010 FOR N=1 TO 1000
1020 function to be timed
1030 NEXT N
1040 T2=TI
1050 PRINT "LOOP TIME=  "
     ;(T2-T1)/60
1060 GOTO 1000
```

will print

```
LOOP TIME= n
LOOP TIME= n
.
.
.
```

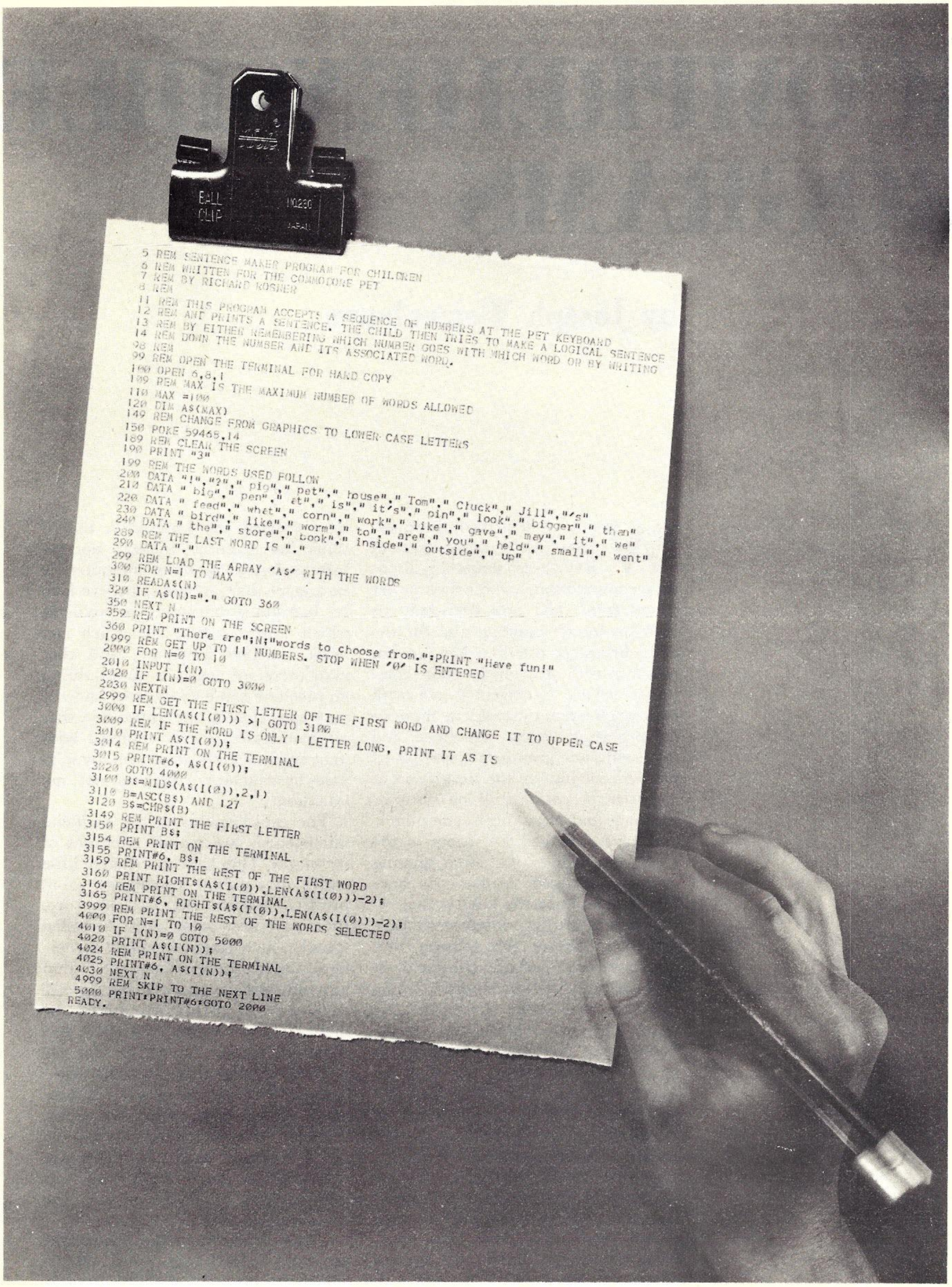
until the stop key is depressed. *N* is the number of seconds required for the *FOR. . .NEXT* loop, and the function to be timed to loop *1000* times. Using this program, the times obtained for the program with different lines *1020* are as shown in table 1.

Since its arrival, the PET has become our family's favorite toy, outpolling even the television in popularity. It's more than a toy, though. We have just managed to add hard-copy capability. So as we familiarize ourselves with the PET's habits and behavior and add more I/O, we expect it to be doing a lot of serious and useful work around the house. ▼

The Pet Talks Hard Copy for the First Time

The program through which our PET spoke its first not-so-halting printed words accepts numbers from the keyboard and generates a sentence. Each number corresponds to a specific word. As a child plays with the keyboard, he generates and changes sentence structure, developing verbal skills.

The hard copy was made on a GE Termi Net 300, using an adapter that converts the parallel data on the IEEE 488 bus to serial data for an RS-232 printer. For more information on this adapter, contact Richard Rosner, 150 Pocono Road, Brookfield, CT 06804.



```
5 REM SENTENCE MAKER PROGRAM FOR CHILDREN
6 REM WRITTEN FOR THE COMMODORE PET
7 REM BY RICHARD ROSNER
8 REM
11 REM THIS PROGRAM ACCEPTS A SEQUENCE OF NUMBERS AT THE PET KEYBOARD
12 REM AND PRINTS A SENTENCE. THE CHILD THEN TRIES TO MAKE A LOGICAL SENTENCE
13 REM BY EITHER REMEMBERING WHICH NUMBER GOES WITH WHICH WORD OR BY WRITING
14 REM DOWN THE NUMBER AND ITS ASSOCIATED WORD.
98 REM
99 REM OPEN THE TERMINAL FOR HARD COPY
100 OPEN 6,8,1
109 REM MAX IS THE MAXIMUM NUMBER OF WORDS ALLOWED
110 MAX =100
120 DIM A$(MAX)
149 REM CHANGE FROM GRAPHICS TO LOWER CASE LETTERS
150 POKE 59465,14
189 REM CLEAR THE SCREEN
190 PRINT "3"
199 REM THE WORDS USED FOLLOW
200 DATA "!","?"," pig"," pet"," house"," Tom"," Cluck"," Jill","/s"
210 DATA " big"," pen"," at"," is"," it's"," pin"," look"," bigger"," than"
220 DATA " feed"," what"," corn"," work"," like"," gave"," may"," it"," we"
230 DATA " bird"," like"," worm"," to"," are"," you"," held"," small"," went"
240 DATA " the"," store"," book"," inside"," outside"," up"
289 REM THE LAST WORD IS "."
290 DATA "."
299 REM LOAD THE ARRAY 'A$' WITH THE WORDS
300 FOR N=1 TO MAX
310 READA$(N)
320 IF A$(N)="." GOTO 360
350 NEXT N
359 REM PRINT ON THE SCREEN
360 PRINT "There are";N;"words to choose from.":PRINT "Have fun!"
1999 REM GET UP TO 11 NUMBERS. STOP WHEN '0' IS ENTERED
2000 FOR N=0 TO 10
2010 INPUT I(N)
2020 IF I(N)=0 GOTO 3000
2030 NEXTN
2999 REM GET THE FIRST LETTER OF THE FIRST WORD AND CHANGE IT TO UPPER CASE
3000 IF LEN(A$(I(0))) >1 GOTO 3100
3009 REM IF THE WORD IS ONLY 1 LETTER LONG, PRINT IT AS IS
3010 PRINT A$(I(0));
3014 REM PRINT ON THE TERMINAL
3015 PRINT#6, A$(I(0));
3020 GOTO 4000
3100 B$=MID$(A$(I(0)),2,1)
3110 B=ASC(B$) AND 127
3120 B$=CHR$(B)
3149 REM PRINT THE FIRST LETTER
3150 PRINT B$;
3154 REM PRINT ON THE TERMINAL
3155 PRINT#6, B$;
3159 REM PRINT THE REST OF THE FIRST WORD
3160 PRINT RIGHT$(A$(I(0)).LEN(A$(I(0)))-2);
3164 REM PRINT ON THE TERMINAL
3165 PRINT#6, RIGHT$(A$(I(0)),LEN(A$(I(0)))-2);
3999 REM PRINT THE REST OF THE WORDS SELECTED
4000 FOR N=1 TO 10
4010 IF I(N)=0 GOTO 5000
4020 PRINT A$(I(N));
4024 REM PRINT ON THE TERMINAL
4025 PRINT#6, A$(I(N));
4030 NEXT N
4999 REM SKIP TO THE NEXT LINE
5000 PRINT:PRINT#6:GOTO 2000
READY.
```

Photograph by Thomas Hall

INCOMPREHENSIBLE PROGRAMS

by Joseph Weizenbaum

We have in preceding discussions (see previous issues of *ROM*) seen something about what computers are, where their power comes from, and how they may be used for model building and for the embodiment of theories. My concern has been with attempts to make computers behave intelligently, not with their application to mundane numerical problems. I have described and discussed some prominent research on problem solving, on the simulation of cognitive processes, and on natural-language understanding by computers, and have described the visions of the artificial intelligentsia, often quoting its acknowledged leaders. I have hardly mentioned the failures that researchers in artificial intelligence have suffered, because the failures that punctuate every research effort are not necessarily grounds for despair; to the contrary, they often enrich the soil from which better ideas later spring. Besides, the fact that something has not yet been done, or even that an attempt to do it failed, does not demonstrate that it cannot be done. In attempting to avoid both trivial and sterile arguments—for example, arguments about whether computers can "in principle" be made to perform this or that specific task—I may even have created the impression that I think the potential (though not yet fully exploited) power of computers to be greater than, in fact, I believe it to be. In arguing that there are problems which confront man but which can never confront machines, and that man therefore comes to know things no machine can ever come to know, I may have created the impression that all problems that may confront both man and machine are potentially solvable by machines. Such is not my intention.

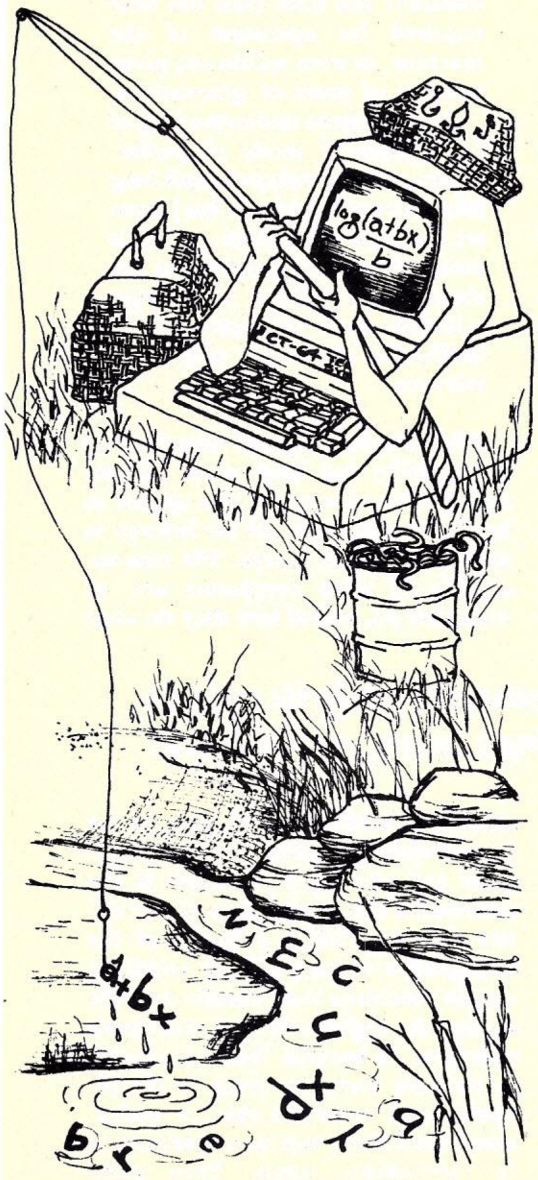
The achievements of the artificial intelligentsia are mainly triumphs of technique. They have contributed little either to cognitive psychology or to practical problem solving. To be sure, there have been what might be called spinoffs, such as refinements in higher-level programming languages, that were initiated by artificial-intelligence concerns and that have entered the mainstream of computer science. But these are hardly the results that the artificial intelligentsia has been forecasting for the "visible future" all these many years. With few exceptions, there have been no results, from over twenty years of artificial-intelligence research, that have found their way into industry generally or into the computer industry in particular.

Two exceptions are the remarkable programs DENDRAL and MACSYMA that exist at Stanford University and at M.I.T., respectively. Both these

---

From *Computer Power and Human Reason,* Copyright © 1976 by W. H. Freeman and Co.

Illustrated by Cindy Hain



programs perform highly technical functions whose discussion is far beyond the scope of this book. But a few words can be said about them.

DENDRAL interprets outputs of mass spectrometers, instruments used for analyses of chemical molecules. In ordinary practice, chemists in post-doctoral training are employed to deduce the chemical structures of molecules given to this instrument from the so-called mass spectra it produces. Their problem is somewhat analogous to that of reconstructing the life of a prehistoric village from the remains uncovered by archeologists. There is, however, an important difference between the two problems: there exists a theory of mass spectrometry; that is, it is known how the instrument generates its output given a particular chemical for analysis. One can therefore evaluate a proffered solution by deducing from the theory what spectrum the instrument would produce if the chemical were what the tentative solution suggests it is. Unfortunately, limitations of precision intervene to make this process of inverse evalutation somewhat less than absolutely exact. Still, the analyst is in a better position than the archeologist, who has no strong methods for verifying his hypotheses. Stated in general terms, then, DENDRAL is a program that analyzes mass spectra and produces descriptions of the structures of molecules that, with very high probability, gave rise to these spectra. The program's competence equals or exceeds that of human chemists in analyzing certain classes of organic molecules.

MACSYMA is, by current standards, an enormously large program for doing symbolic mathematical manipulations. It can manipulate algebraic expressions involving formal variables, functions, and numbers. It can differentiate, integrate, take limits, solve equations, factor polynomials, expand functions in power series, and so on. It does all these things symbolically, not numerically. Thus, for example, given the problem of evaluating

$$\int \frac{dx}{a+bx},$$

it will produce

$$\frac{\log\,(a+bx)}{b}.$$

Of course, if it is given numerical values for all the variables involved, it will give the numerical value of the whole expression, but that is for it a relatively trivial task. Again, the technical details involved are beyond the scope of this discussion. What is important here is that, just as for DENDRAL, there exist strong theories about how the required transformations are to be made. Most importantly, especially for symbolic integration, it is possible (by differentiating) to check whether or not a proffered solution is in fact a solution, and, for integration, the test is absolute. And just as for DENDRAL, MACSYMA's task is one that is normally accomplished only by highly trained specialists.

These two programs owe a significant debt to the artificial-intelligence movement. They both use heuristic problem-solving methods in two distinct ways. First, when the design of these programs was initiated, the theories on which they are now based were not sufficiently well-formed to be modeled in terms of effective procedures. Yet people accomplished the required tasks. An initial problem was therefore to extricate from experts the heuristics they used in doing what they did. The initial versions of these programs were a mixture of algorithms incorporating those aspects of the problems that were well understood, and encodings of whatever heuristic techniques could be gleaned from experts. As the work progressed, however, the programs' heuristic components became increasingly well understood and hence convertible to enrichments of the relevant theories. Both programs were thus gradually modified until they became essentially completely theory-based. Second, heuristic methods were and continue to be used in both programs for reasons of efficiency. Both programs generate subproblems which, though in principle solvable by straightforward algorithmic means, yield more easily after they are classified as being amenable to solution by some special function and are then turned over to that function for solution. The development and refinement of both uses of heuristic methods, as well as many of the methods themselves, are products of artificial-intelligence research.

Decisions are made with the aid of, or entirely by, computers whose programs no one any longer knows or understands.

These two programs are distinguished from most other artificial-intelligence programs precisely in that they rest solidly on deep theories. The principal contributor of the theoretical underpinnings of DENDRAL was Joshua Lederberg, the geneticist and Nobel laureate, and MACSYMA's theoretical base is principally the work of Professor Joel Moses of M.I.T., an extremely talented and accomplished mathematician.

There are, of course, many other important and successful applications of computers. Computers, for example, control entire petroleum-refining plants, navigate spaceships, and monitor and largely control the environments in which astronauts perform their duties. Their programs rest on mathematical control theory and on firmly established physical theories. Such theory-based programs enjoy the enormously important advantage that, when they misbehave, their human monitors can detect that their performance does not correspond to the dictates of their theory and can diagnose the reason for the failure from the theory.

But most existing programs, and especially the largest and most important ones, are not theory-based in this way. They are heuristic, not necessarily in the sense that they employ heuristic methods internally, but in that their construction is based on rules of thumb, strategems that appear to "work" under most foreseen circumstances, and on other ad-hoc mechanisms that are added to them from time to time.

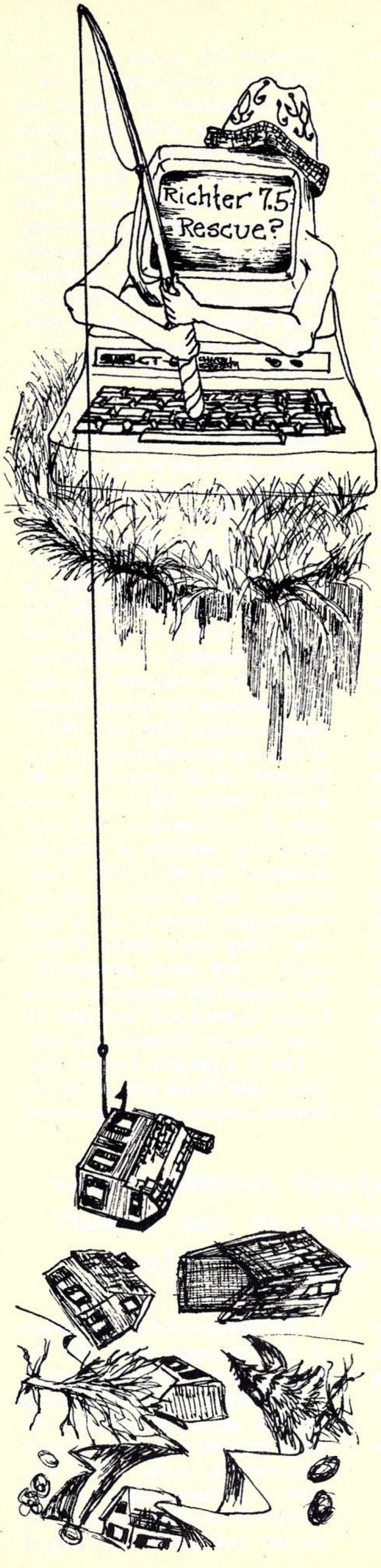
My own program, ELIZA, was of precisely this type. So is Winograd's language-understanding system and, all pretensions to the contrary notwithstanding, Newell and Simon's GPS. What is much more important, however, is that almost all the very large computer programs in daily use in industry, in government, and in the universities are of this type as well. These gigantic computer systems have usually been put together (one cannot always use the word "designed") by teams of programmers, whose work is often spread over many years. By the time these systems come into use, most of the original programmers have left or turned their attention to other pursuits. It is precisely when such systems begin to be used that their inner workings can no longer be understood by any single person or by a small team of individuals.

Norbert Wiener, the father of cybernetics, foretold this phenomenon in a remarkably prescient article published almost fifteen years ago. He said there:

> It may well be that in principle we cannot make any machine the elements of whose behavior we cannot comprehend sooner or later. This does not mean in any way that we shall be able to comprehend these elements in substantially less time than the time required for operation of the machine, or even within any given number of years or generations.
>
> An intelligent understanding of [a machine's] mode of performance may be delayed until long after the task which [it has] been set has been completed.... This means that, though machines are theoretically subject to human criticism, such criticism may be ineffective until long after it is relevant.

What Norbert Wiener described as a possibility has long since become reality. The reasons for this appear to be almost impossible for the layman to understand or to accept. His misconception of what computers are, of what they do, and of how they do what they do is attributable in part to the pervasiveness of the mechanistic metaphor and the depth to which it has penetrated the unconscious of our entire culture. This is a legacy of the imaginative impact of the relatively simple machines that transformed life during the eighteenth and nineteenth centuries. It became "second nature" to virtually everyone living in the industrialized countries that to understand something was to understand it in mechanistic terms. Even great

They have reduced reason to only its role in the domination of things, man, and nature.



scientists of the late nineteenth century subscribed to this view. Lord Kelvin (1824-1907) wrote: "I never satisfy myself until I can make a mechanical model of a thing. If I can make a mechanical model, I can understand it. As long as I cannot make a mechanical model all the way through, I cannot understand it." An expression of the corresponding modern sentiment is Minsky's belief that to understand music and "highly meaningful pictures" means to be able to write computer programs that can generate these things. But whereas Minsky deeply understands that computers are not machines to be equated with the mechanisms Kelvin knew, the layman understands just the contrary. To him computers and computer programs are "mechanical" in the same simple sense as steam engines and automobile transmissions.

This belief—and it is virtually universal among laymen—is reinforced by the slogan often repeated by computer scientists themselves that "unless a process is formulated with perfect precision, one cannot make a computer do it." This slogan is true, however, only under a very strict and most unusual interpretation of what it means to "formulate a process." If one were to throw a random pattern of bits into a computer's store, for example, and set the computer to interpret it as a program, then, assuming it would "work" at all, that bit pattern would be a "formulation" of some process. But program formulation is understood in normal discourse to mean that some agent (probably human) organized what is to become a computer program before giving it to the computer. The layman, having heard the slogan in question, believes that the very fact that a program runs on a computer guarantees that some programmer has formulated and understands every detail of the process which it embodies.

But his belief is contradicted by fact. A large program is, to use an analogy of which Minsky is also fond, an intricately connected network of courts of law, that is, of subroutines, to which evidence is transmitted by other subroutines. These courts weigh (evaluate) the data given them and then transmit their judgments to still other courts. The verdicts rendered by these courts may, indeed, often do, involve decisions about what court has "jurisdiction" over the intermediate results then being manipulated. The programmer thus cannot even know the path of decisionmaking within his own program, let alone what intermediate or final results it will produce. Program formulation is thus rather more like the creation of a bureaucracy than like the construction of a machine of the kind Lord Kelvin may have understood. As Minsky puts it:

> The programmer himself state[s] . . . "legal" principles which permit . . . "appeals," he may have only a very incomplete understanding of when and where in the course of the program's operation these procedures will call on each other. And for a particular "court," he has only a sketchy idea of only some of the circumstances that will cause it to be called upon. In short, once past the beginner level, . . . programmers write—not "sequences" [of instructions]—but specifications for the individuals of little societies. Try as he may he will often be unable fully to envision in advance all the details of their interactions. For that, after all, is why he needs the computer.

Minsky goes on to make the following enormously important observations:

> When a program grows in power by an evolution of partially understood patches and fixes, the programmer begins to lose track of internal details, loses his ability to predict what will happen, begins to hope instead of know, and watches the results as though the program were an individual whose range of behavior is uncertain.
>
> This is already true in some big programs . . . . it will soon be much more acute . . . . large heuristic programs will be developed and modified by several programmers, each testing them on different examples from different [remotely located computer] consoles and inserting advice indedently. The program will grow in effectiveness, but no one of the programmers will understand it all. (Of course, this won't always be successful—the interactions might make it get worse, and no one might be able to fix it again!)

> Now we see the real trouble with statements like "it only does what its programmer told it to do." There isn't any one programmer.

We do not understand, to hark back to an earlier point for a moment, how a program of the kind Minsky here describes—one that, say, composes "great" music—helps us to "understand" music when the program itself is beyond our understanding.

But, more importantly, if the program has outrun the understanding of the agents who created it, what can it mean for it to "grow in effectiveness," or, for that matter, to "get worse?" As teachers (and we are all teachers) we, of course, constantly hope that those we instruct will grow in effectiveness in their dealings with whatever it is that is the subject of our tutelage. And we do not usually require of ourselves that we "understand" the processes whose growth we mean to encourage in our students, that is, that we understand them in the same way as, say, we understand the workings of a clock. Moreover, we do invest our hopes in our students and rely on them and trust them.

The computer has become an instrument for the destruction of history.

It is undoubtedly this kind of trust that Minsky urges us to invest in complex artificial-intelligence programs that grow in effectiveness but which come to be beyond our understanding. His advice is entirely reasonable if it applies to programs for which we have performance measures that enable us to tell, and tell in sufficient time, when these programs are operating outside an acceptable range of behavior or when, for any reason, they no longer deserve our trust. As we observed earlier, programs that are fundamentally models of well-understood theories fall into this class, even if, as may happen, no group of programmers commands a detailed understanding of the innards of the programs themselves. So too do programs whose drift away from performance criteria can be detected by observations of their moment-to-moment behavior—providing, of course, first, that someone responsible is watching, and, second, that he can intervene in time to avoid disaster. But there now exist many important programs that are very large and complex and that do not meet these criteria.

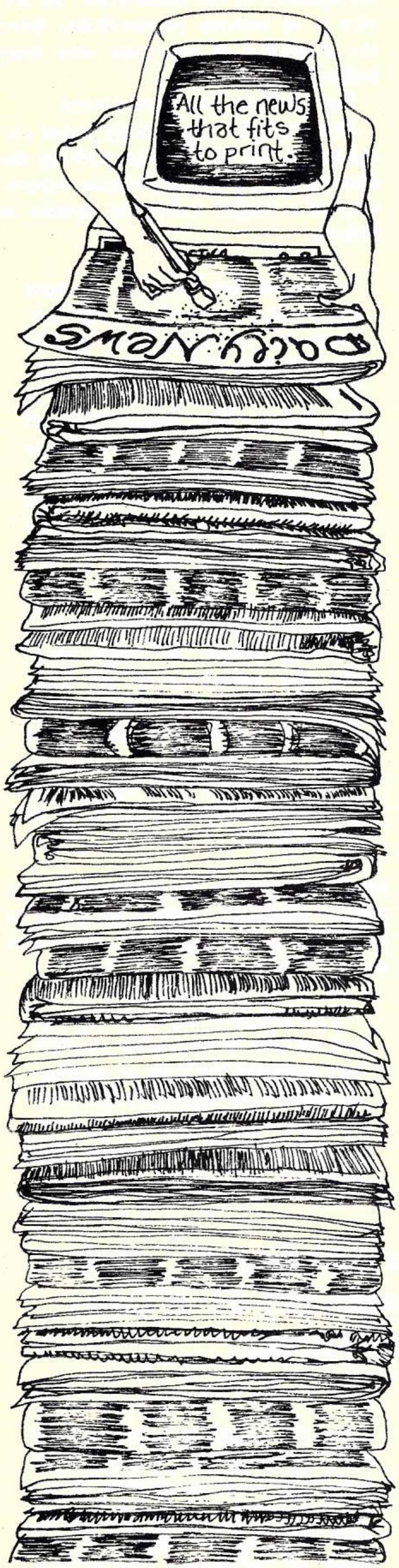
Minsky's account, which is entirely accurate, is thus of the utmost importance. It tells us that the condition Norbert Wiener described as a possibility in 1960 has quickly become and is now a reality. Minsky's words, moreover, take on a special importance because they were written by one of the chief architects and spokesmen for artificial intelligence, and were intended to correct the fuzzy thinking of humanists by extolling the power of computers, not their limitations.

Our society's growing reliance on computer systems that were initially intended to "help" people make analyses and decisions, but which have long since both surpassed the understanding of their users and become indispensable to them, is a very serious development. It has two important consequences. First, decisions are made with the aid of, and sometimes entirely by, computers whose programs no one any longer knows explicitly or understands. Hence no one can know the criteria or the rules on which such decisions are based. Second, the systems of rules and criteria that are embodied in such computer systems become immune to change, because, in the absence of a detailed understanding of the inner workings of a computer system, any substantial modification of it is very likely to render the whole system inoperative and possibly unrestorable. Such computer systems can therefore only grow. And their growth and the increasing reliance placed on them is then accompanied by an increasing legitimation of their "knowledge base."

The enormous computer systems in the Pentagon and elsewhere in our culture have, in a very real sense, no authors.

Professor Philip Morrison of M.I.T. wrote a poignant parable on this theme:

> On the wall of my office is a world map, computer-plotted and therefore not as beautiful as a draftsman would manage. On it are bold outlines, in eight or ten thousand dots, of the huge plates that make up the crust of the earth, which, when they spread apart or touch together or ride one over the other, generate most, perhaps nearly all, substantial earthquakes. The map embodies that realization, for its dotted outlines of plates were made by thousands of earthquake foci.
>
> The curious part is this: the seismologists responsible for the map say, somewhat apologetically, that since their own recordings of earthquakes were in one standard format which could easily be told to the machine so as to locate the dots on the map, they could use only their own data. They knew, to be sure, that seismology is much older than this decade, but the effort to try to connect the past to a standard coordinate system, to put in readable form into their computer the vast and diverse literature from 1840 until 1961—all this was beyond them. So they dropped out all reference to the science before 1961, and used only the earthquakes their own world-wide network of detectors recorded from 1961 to 1967. That, however, was as many as all the earthquakes recorded up to that time. They lost a factor of two, which is not much statistically; they gained the advantage of not having to read and interpret all those obscure German journals.
>
> This is a parable for the computer. Like all parables, it has an internal tension: it gives something to the enemies and to the friends of the computer alike. For the friends it is patent that this superb collection of epicenters delineating tectonic plates is probably the single greatest accomplishment of such synoptic study. For an outsider, it is fascinating to see the outline of the rifts and joints. At last we understand something of


All the news that fits to print.
Daily News

the earth in the large. At the same time, so cavalier a dismissal of the entire history of a science is breathtaking.

The lesson is quite plain: nobody, not the most single-minded proponent of computer data processing, would say that it all began in 1961, even if our modern compatible data began then. The past was an indespensable prologue; it saw the formation of concepts, the development of techniques, the introduction of instruments, the idea of systematic recording, and so on. All this showed the way, without which I am sure the Coast and Geodetic Survey and its friends would not have been able to produce so beautiful a map.

The computer has thus begun to be an instrument for the destruction of history. For when society legitimates only those "data" that are "in one standard format" and that "can easily be told to the machine," then history, memory itself, is annihilated. The *New York Times* has already begun to build a "data bank" of current events. Of course, only those data that are easily derivable as by-products of typesetting machines are admissable to the system. As the number of subscribers to this system grows, and as they learn more and more to rely on "all the news that [was once] fit to print," as the *Times* proudly identifies its editorial policy, how long will it be before what counts as fact is determined by the system, before all other knowledge, all memory, is simply declared illegitimate? Soon a supersystem will be built, based on the *New York Times*' data bank (or one very like it), from which "historians" will make inferences about what "really" happened, about who is connected to whom, and about the "real" logic of events. There are many people now who see nothing wrong in this.

We do not have to go to prospective systems to fill out Morrison's parable. In the recent American war against Viet Nam, computers operated by officers who had not the slightest idea of what went on inside their machines effectively chose which hamlets were to be bombed and what zones had a sufficient density of Viet Cong to be "legitimately" declared free-fire zones, that is, large geographical areas in which pilots had the "right" to kill every living thing. Of course, only "machine readable" data, that is, largely targeting information coming from other computers, could enter these machines. And when the American President decided to bomb Cambodia and to keep that decision secret from the American Congress, the computers in the Pentagon were "fixed" to transform the genuine strike reports coming in from the field into the false reports to which government leaders were given access. George Orwell's Ministry of Information had become mechanized. History was not merely destroyed, it was recreated. And the high government leaders who felt themselves privileged to be allowed to read the secret reports that actually emerged from the Pentagon's computers of course believed them. After all, the computer itself had spoken. They did not realize that they had become their computer's "slaves," to use Admiral Moorer's own word, until the lies they instructed their computers to tell others ensnared them, the instructors, themselves.

In modern warfare it is common for the soldier, say, the bomber pilot, to operate at an enormous psychological distance from his victims. He is not responsible for burned children because he never sees their village, his bombs, and certainly not the flaming children themselves. Modern technological rationalizations of war, diplomacy, politics, and commerce (such as computer games) have an even more insidious effect on the making of policy. Not only have policy makers abdicated their decision-making responsibility to a technology they do not understand—though all the while maintaining the illusion that they, the policy makers, are formulating policy questions and answering them—but responsibility has altogether evaporated. Not only does the most senior admiral of the United States Navy, in a rare moment of insight, perceive that he has become "a slave to these damned computers," that he cannot help but base his judgments on "what the computer says," but no human is responsible at all for the computer's output. The enormous computer systems in the Pentagon and their counterparts elsewhere in our culture have, in a very real sense, no authors. Thus they do not admit of any questions of right or wrong, of justice, or of any theory with which one can agree or disagree. They provide no basis on which "what the machine says" can be challenged. My father used to invoke the ultimate authority by saying to me "It stands written!" But then I read what stood written, imagine a human author, infer his values, and finally agree or disagree with him. Computer systems do not admit of exercises of imagination that may ultimately lead to authentic human judgment.

They didn't realize they'd become slaves until the lies their computers told others ensnared them too.

No wonder that men who live day in and day out with machines to which they perceive themselves to have become slaves begin to believe that men are machines, that, as an important scientist once put it:

> It is possible to look on Man himself as a product of...an evolutionary process of developing robots, begotten by simpler robots, back to the primordial slime; ... his ethical conduct [is] something to be interpreted in terms of the circuit action of...Man in his environment—a Turing machine with only two feedbacks determined, a desire to play and a desire to win.

One would expect that large numbers of individuals, living in a society in which anonymous, hence irresponsible, forces formulate the large questions of the day and circumscribe the range of possible answers, would experience a kind of impotence and fall victim to a mindless rage. And surely we see that expectation fulfilled all around us, on university campuses and in factories, in homes and offices. Its manifestations are workers' sabotage of the products of their labor, unrest and aimlessness among students, street crime, escape into drug-induced dream worlds, and so on. Yet an alternative response is also very pervasive; as seen from one perspective, it appears to be resignation, but from another perspective it is what Erich Fromm long ago called "escape from freedom."

The "good German" in Hitler's time could sleep more soundly because he "didn't know" about Dachau. He didn't know, he told us later, because the highly organized Nazi system kept him from knowing. (Curiously, though, I, as an adolescent in that same Germany, knew about Dachau. I thought I had reason to fear it.) Of course, the real reason the good German didn't know is that he never felt it to be his responsibility to ask what had happened to his Jewish neighbor whose apartment suddenly became available. The university professor whose dream of being promoted to the status of Ordinarius was suddenly fulfilled didn't ask how his precious chair had suddenly become vacant. Finally, all Germans became victims of what had befallen them.

Today even the most highly placed managers represent themselves as innocent victims of a technology for which they accept no responsibility and which they do not even pretend to understand. (One must wonder, though, why it never occurred to Admiral Moorer to ask what effect the millions of tons of bombs the computer said were being dropped on Viet Nam were having.) The American Secretary of State, Dr. Henry Kissinger, while explaining that he could hardly have known of the "White House horrors" revealed by the Watergate investigation, mourned over "the awfulness of events and the tragedy that has befallen so many people."

The tragedy so described had action, but no actors. Only "events" were "awful"—not individuals or officials. In this lifeless setting, the mockery of law and the deceit of the people had not been rehearsed and practiced: they had simply "befallen."

Policy makers have abdicated their decision-making responsibility to a technology they do not understand.

The myth of technological and political and social inevitability is a powerful tranquilizer of the conscience. Its service is to remove responsibility from the shoulders of everyone who truly believes in it.

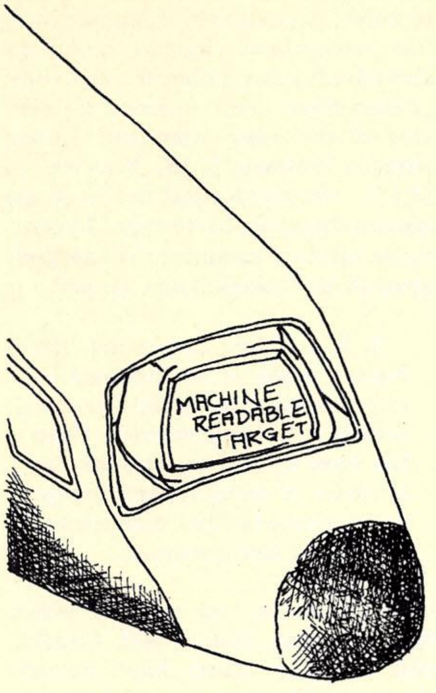
But, in fact, there *are* actors!

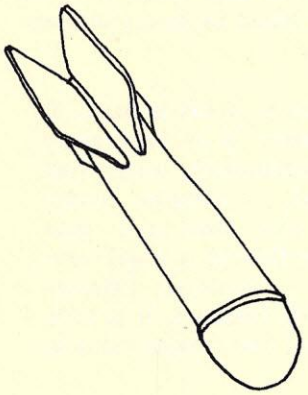
For example, a planning paper circulated to the faculty and staff by the director of a major computer laboratory of a major university speaks as follows:

> Most of our research has been supported, and probably will continue to be supported, by the Government of the United States, the Department of Defense in particular. The Department of Defense, as well as other agencies of our government, is engaged in the development and operation of complex systems that have a very great destructive potential and that, increasingly, are commanded and controlled through digital computers. *These systems are responsible,* in large part, for the maintenance of what peace and stability there is in the world, and at the same time they are capable of unleashing destruction of a scale that is almost impossible for man to comprehend.

Note that systems are responsible, not people. Anyway, so much for a nod to their destructive potential; now on to the real concerns:

> The crucial role of computers can be seen more vividly in military applications than in applications in the non-military sectors of society, but most of us have thought enough about the progressively increasing dependence upon computers in commerce and industry to project a picture in which the very functioning of society depends upon an orderly and meaningful execution of billions of electronic instructions

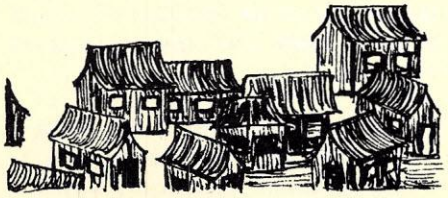




> every second. . . . there will be large-scale systems with millions of words of fast random-access memory, capable of tens of millions of instructions per second, in organizations of every kind. Most of these computers will be linked together in complexes of networks through which they will have access (governed by control mechanisms derived from what some of us are doing now) to all the information there is about everthing and everybody. *And there is no stemming this trend in computer development*. . . .
>
> In mastering the programming and control of computers, *we* especially could play a critical role. It may well be that no other organization is able to play this role as we are, yet no more important role may exist in science and engineering today.
>
> The importance of the role stems, as has been noted, from the fact that *the computer has been incorporating itself,* and will surely continue to incorporate itself, into most of the functions that are fundamental to the support, protection, and development of our society. Even now, *there is no turning back,* and in a few years it will be clear that we are as vitally dependent upon the informational processing of our computers as upon the growth of grain in the field and the flow of fuel from the well.

There is not the slightest hint of a question as to whether we want this future. It is simply coming. We are helpless in the face of a tide that will, for no reason at all, not be stemmed. There is no turning back. Even the question is not worth discussing.

> There are many facets to the general problem of mastery and control. They range from essentially philosophical problems that concern meaning and intentions and the establishment of conformity between a plan and the actual behavior of a complex system to the almost purely technical problem of finding bugs in subroutines.



(In computer jargon, a "bug" is a programming error.) Notice the parochial, that is technological, view of philosophy that is displayed here. But to go on:

> One should not be satisfied with methods of programming that let bugs get into programs. It is probable that the best way to eliminate bugs is to devise bug-free methods of programming. Nevertheless, debugging [the elimination of bugs from programs] should be in the focus of the research effort undertaken to master programming. The reason is that research on debugging will yield insight into many problems in the formulation and expression of human intentions. It is not the mere coding of a program formulation of problem and solution. As programming is mastered, there will be a continual enlarging of the scope of problem solving, a widening of the universe of discourse. The goal should be a method of programming that is as free of bugs and glitches in these higher levels as it is in the lower ones.

What is so remarkable about this is that the main—indeed, the only—impediment to "problem solving," even at "these higher levels," is seen to be entirely a matter of technical errors. There are no genuine conflicts in society. Once we understand "human intentions," itself a technical problem, all else is technique.

> Almost certainly, computers are freer of error in doing whatever they do than people are. If we can create a program-writing program, therefore, we should be a long step along the way to bug-free software. It is important, however, not to gloss over the problems that arise at the interface between the human statement of a problem and the computer understanding of that problem. The computer is unlikely to prepare a proper program unless its comprehension of the problem is entirely correct. . . .
>
> A possibly important approach to the mastering of programming and debugging is based on modeling. . . . In the beginning, the human programmer uses such models as he has available as aids. Toward the end, the models, combined into one comprehensive

> model, are doing most of the programming and debugging, but the human programmer is still in the picture to supervise or help out or provide heuristic guidance or whatever. Eventually, if the effort is successful, the model becomes the automatic programmer....
>
> Only a year or two ago, it was necessary to put quotation marks around the word "knowledge" whenever it was used in such a context as this...but [within a rather small circle of computer scientists] there is a consensus that we have reached the threshold beyond which one can think of computers as having knowledge and using it effectively and meaningfully in ways analogous, and probably in due course superior to, the ways in which human beings use knowledge....

Instrumental reason has made out of words a fetish surrounded by black magic.

> Conversations with and among [our] faculty give rise to a strong feeling of convergence in a new direction. Many seem to sense a common set of priorities. Putting these feelings into words has required much sober thought, reflection and many extended discussions.
>
> The convergence of direction...involves making computers not only easy to use but, as has been stressed here, *trustworthy*....
>
> [Our] unique resources are better utilized if totally harnessed to assure that the computer dominated future is one *we* wish. Perhaps [we] *only* [are] in a position to see this important goal attained.

The author of this document—he is, by the way, not anyone mentioned elsewhere in this article—is merely proposing the implementation of the program so often trumpeted by other spokesmen for technological optimism. What he writes is entirely consistent with, for example, H. A. Simon's forecast made in 1960 that:

> Within the very near future—much less than twenty-five years—we shall have the technical capability of substituting machines for any and all human functions in organizations. Within the same period, we shall have acquired an extensive and empirically tested theory of human cognitive processes and their interaction with human emotions, attitudes, and values.

Nor is the "optimism" displayed here related only to computers. Professor B. F. Skinner, the leader of behaviorism in psychology, and of whom it is often said that he is the most influential psychologist alive today, wrote recently,

> The disastrous results of common sense in the management of human behavior are evident in every walk of life, from international affairs to the care of a baby, and we shall continue to be inept in all these fields until a scientific analysis clarifies the advantages of a more effective technology.
>
> In the behavioristic view, man can now control his own destiny because he knows what must be done and how to do it.

That last sentence cannot be read as meaning anything other than, "I, B. F. Skinner, know what must be done and how to do it," just as the last sentence of the planning paper I quoted cannot mean anything other than that the "we," whose job it is to harness our resources to assure the computer-dominated future that "we" wish, are the members of the rather small circle of computer scientists within which "we" can speak candidly and without the use of euphemistic quotation marks. It is significant that both sentences are the closing sentences of their documents. They contain, obviously, the final and most important message.

The myth of technological, political, and social inevitability is a powerful tranquilizer of the conscious.

But the technological messiahs, who, because they find it impossible to trust the human mind, feel compelled to build "trustworthy" computers that will comprehend human intentions and solve human problems, have competitors from other quarters as well. One of the most prominent among them is Professor J. W. Forrester of M.I.T., the intellectual father of the systems-dynamics movement. In testimony before a committee of the Congress of the United States, he said,

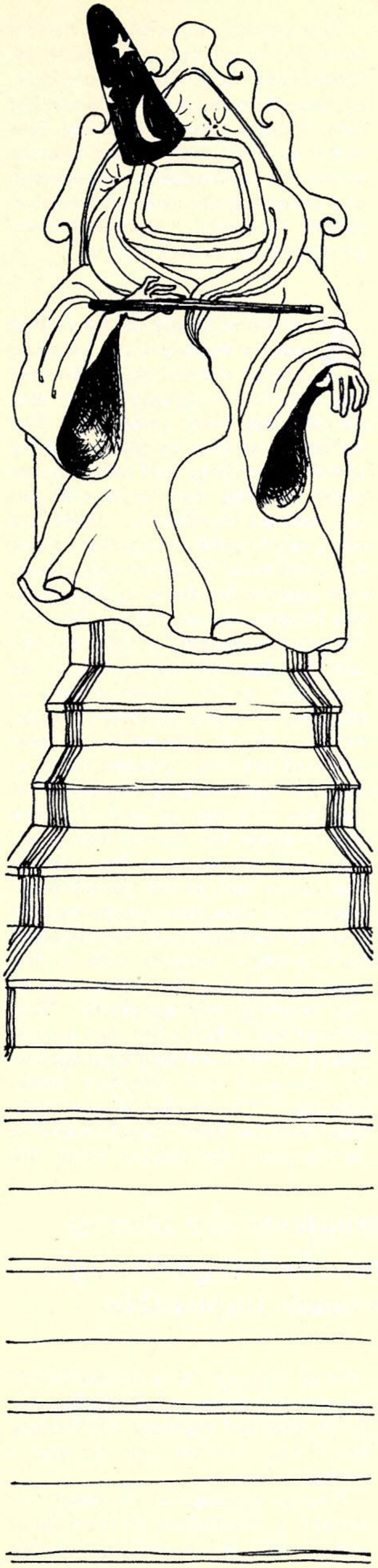
> It is my basic theme that the human mind is not adapted to interpreting how social systems behave.... Until recently there has been no way to estimate the behavior of social systems except by contemplation, discussion, argument, and guesswork.

In other words, the ways in which Plato, Spinoza, Hume, Mill, Gandhi, and so many others have thought about social systems are obviously inferior to the way of systems analysis. The trouble is that these ways of thinking are based on mental models. And:

> The mental model is fuzzy. It is incomplete. It is imprecisely stated. Furthermore, within one individual, a mental model changes with time and even during the flow of a single conversation.... Goals are different and are left unstated. It is little wonder that compromise takes so long.

Clearly, goals must be fixed, hence mental models too, else how can we determine the operators (to use GPS language) that are to be applied to the objects we wish to transform into "desired objects?" And the fuzziness of mental models is, Forrester observes, largely due to the fuzziness of human language itself. That must be repaired too.

> Computer models differ from mental models in important ways. The computer models are stated



> explicitly. The "mathematical" notation that is used for describing the model is unambiguous. It is a language that is clearer and more precise than the spoken languages like English or French. Computer model language is a simpler language. Its advantage is in the clarity of meaning and the simplicity of the language syntax. The language of a computer model can be understood by almost anyone, regardless of educational background. Furthermore any concept and relationship that can be clearly stated in ordinary language can be translated into computer model language.

One has to wonder why it is that ordinary language, what with all its dysfunctional properties, survives at all. And if it is so clear that every concept and relationship can be translated into computer terms, why do the linguists, e.g., Halle, Jakobson, Chomsky, continue to struggle so mightily? And why are there still poets? More to the present point, however, it is simply not true that "almost anyone" can understand the language of, say, Forrester's computer models. The latter have been widely accepted mainly because they were produced by a famous scientist affiliated with a prestigious university, and because their results are "what the computer says." Most ministers of state, labor leaders, and social commentators who have engaged in the "limits of growth" debate could no more read the computer programs which underly the controversy than they can read the equations of quantum physics. But, like Admiral Moorer, they find it useful to "trust" the machine.

Professor Forrester, finally, reassures his audience that "the means are visible" (to him, of course) for beginning to end uncertainty.

> The great uncertainty with mental models is the inability to anticipate the consequences of interactions between parts of a system. This uncertainty is totally eliminated in computer models. Given a stated set of assumptions, the computer traces the resulting consequences without doubt or error.

He goes on to say that, although in our social system there are "no utopias" and no sustainable modes of behavior that are free of pressures and stresses, some possible modes of behavior are more "desirable" than others. And how are these more desirable modes of behavior enabled?

> They seem to be possible only if we have a good understanding of the system dynamics and are willing to endure the self-discipline and pressures that must accompany the desirable mode.

There is undoubtedly some interpretation of the words "system" and "dynamics" which would lend a benign meaning to this observation. But in the context in which these words were spoken, they have the special meaning given them by Forrester. It is then clear that Forrester's message is quite the same as Skinner's and the others': the only way to gain the understanding which alone leads to "desirable modes of behavior" is Forrester-like (or Skinner-like, or GPS-like, and so on) methods of "scientific analysis."

The various systems and programs we have been discussing share some very significant characteristics: they are all, in a certain sense, simple; they all distort and abuse language; and they all, while disclaiming normative content, advocate an authoritarianism based on expertise. Their advocacy is, of course, disguised by their use of rhetoric couched in apparently neutral, jargon-laden, factual language (that is, by what the common man calls "bullshit"). These shared characteristics are, to some extent, separable, but they are not independent of one another.

The most superficial aspects of these system's simplicity—as reflected by their simplistic construction of their subject matters—are immediately visible. Simon, for example, sees man as "quite simple." The "apparent" complexity of his behavior is due to the complexity of his environment. In any event, he can be simulated by a system sensitive to only "a few simple parameters," one that consists of only a few (certainly many fewer than, say, ten thousand) "elementary information processes." The laboratory director I quoted believes that the problem of human intentionality can be usefully attacked by research on computer program-debugging techniques, a belief shared by many of his colleagues. Skinner sees man as essentially a passive product (victim) of his genetic endowment and his history of reinforcing contingencies. The main difference between Skinner's system and GPS appears to be that Skinner is willing to look only at "input-output behavior" (to use computer jargon), whereas the architects of GPS and similar systems feel they can say something about what goes on inside the organism as well. But the philosophical differences between the two attitudes are slight. Forrester sees literally the whole world in terms of feedback loops.

All concepts, ideas, images that can't be put into computer-comprehensible language have lost their function and potency.

> Feedback loops are the fundamental building blocks of systems.... A feedback loop is composed of two kinds of variables, called here rate and level variables. These two kinds of variables are necessary and sufficient.... the level variables are accumulations or integrations....rates of flow cause the levels to change. The levels provide the information inputs to the rate equations which control the flows.
>
> The rate equations are the statements of system policy. They determine how the available information is converted to an action stream.... A rate equation states the discrepancy between the goal and the observed condition. And finally, the rate equation states the action that will result from the discrepancy.

Notice the overlap in language between this statement and Newell and Simon's discussion of problems and problem solving (December, 1977 issue of *ROM*). The latter talk about "present objects" and "desired object," differences between them, operators that reduce these differences, goals, and so on. The difference between their system and Forrester's lies chiefly in the different sets of "elementary information processing" primitives each employs and, of course, in the fact that GPS uses heuristic methods to reduce searches for operators, etc., whereas in Forrester's system everything is explicitly algorithmrized in terms of rate and level variables enmeshed in feedback loops. But the worldviews represented by these two systems are basically the same. And they are very simple.

But these systems are simple in a deeper and more important sense as well. They have reduced reason itself to only its role in the domination of things, man, and, finally, nature.

> Concepts have been reduced to summaries of the characteristics that several specimens have in common. By denoting similarity, concepts eliminate the bother of enumerating qualities and thus serve better to organize the material of knowledge. They are thought of as mere abbreviations of the items to which they refer. Any use transcending auxiliary, technical summarization of factual data has been eliminated as a last trace of superstition. Concepts have become "streamlined," rationalized, labor-saving devices...thinking itself [has] been reduced to the level of industrial processes...in short, made part and parcel of production.

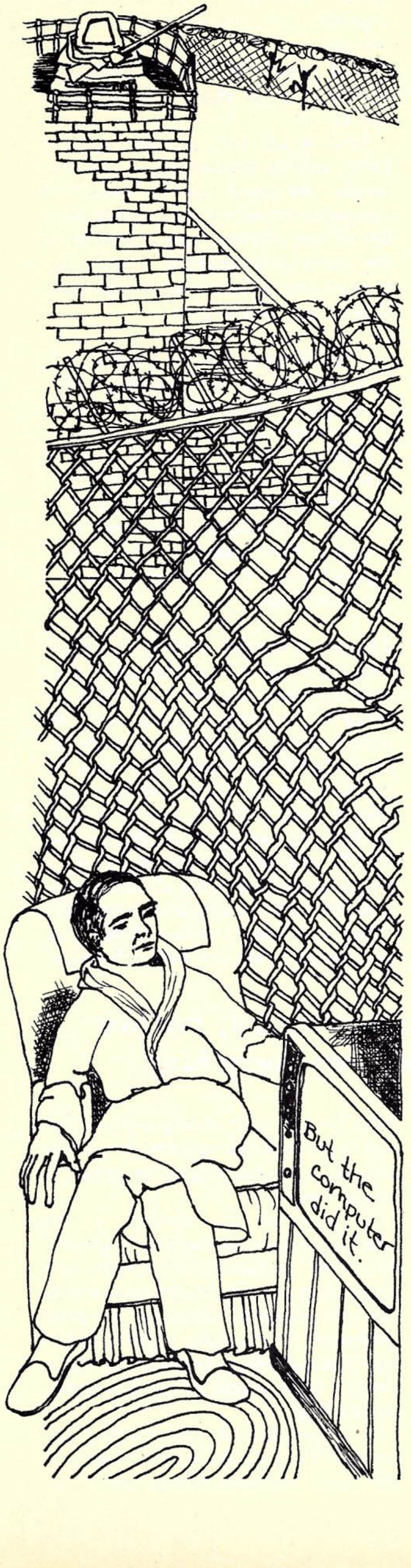
No one who does not know the technical basis of the systems we have been discussing can possibly appreciate what a chillingly accurate account of them this passage is. It was written by the philosopher-sociologist Max Horkeimer in 1947, years before the forces that were even then eclipsing reason, to use Horkeimer's own expression, came to be embodied literally in machines.

When international problems are seen as mere technical problems, the recognition of conflicting interests is made impossible.

This passage, especially in view of when and by whom it was written, informs us once again that the computer, as presently used by the technological elite, is not a cause of anything. It is rather an instrument pressed into the service of rationalizing, supporting, and sustaining the most conservative, indeed, reactionary, ideological components of the current *Zeitgeist.*

As we see so clearly in the various systems under scrutiny, meaning has become entirely transformed into function. Language, hence reason too, has been transformed into nothing more than an instrument for affecting the things and events in the world. Nothing these systems do has any intrinsic significance. There are only goals dictated by tides that cannot be turned back. There are only means-ends analyses for detecting discrepancies between the way things are, the "observed condition," and the way the fate that has befallen us tells us we wish them to be. In the process of adapting ourselves to these systems, we, even the admirals among us, have castrated not only ourselves (that is, resigned ourselves to impotence), but our very language as well. For now that language has become merely another tool, all concepts, ideas, images that artists and writers cannot paraphrase into computer-comprehensible language have lost their function and their potency. Forrester tells us this most clearly—but the others can be seen nodding their agreement: "Any concept and relationship that can be clearly stated in ordinary language can be translated into computer model language." The burden of proof that something has been "stated clearly" is on the poet. No wonder, given this view of language, that the distinction between the living and the lifeless, between man and machine, has become something less than real, at most a matter of nuance!

Corrupt language is very deeply imbedded in the rhetoric of the technological elite. We have already noted


But the computer did it.

the transformation of the meaning of the word "understand" by Minsky into a purely instrumental term. And it is this interpretation of it that, of course, pervades all the systems we have been discussing. Newell and Simon's use of the word "problem" is another example and one just as significant.

During the times of trouble on American university campuses, one could often hear well-meaning speakers say that the unrest, at least on their campuses, was mainly caused by inadequate communication among the university's various constituencies, e.g., faculty, administration, students, staff. The "problem" was therefore seen as fundamentally a communication, hence a technical, problem. It was therefore solvable by technical means, such as the establishment of various "hotlines" to, say, the president's or the provost's office. Perhaps there were communication difficulties; there usually are on most campuses. But this view of the "problem"—a view entirely consistent with Newell and Simon's view of "human problem solving" and with instrumental reasoning—actively hides, buries, the existence of real conflicts. It may be, for example, that students have genuine ethical, moral, and political interests that conflict with interests the university administration perceives itself to have, and that each constituency understands the other's interests very well. Then there is a genuine problem, not a communication difficulty, certainly not one that can be repaired by the technical expedient of hotlines. But instrumental reason converts each dilemma, however genuine, into a mere paradox that can then be unraveled by the application of logic, by calculation. All conflicting interests are replaced by the interests of technique alone.

This, like Philip Morrison's story, is a parable too. Its wider significance is that the corruption of the word "problem" has brought in its train the mystique of "problem solving," with catastrophic effects on the whole world. When every problem on the international scene is seen by the "best and the brightest" problem solvers as being a mere technical problem, wars like the Viet Nam war become truly invevitable. The recognition of genuinely conflicting but legitimate interests of coexisting societies—and such recognition is surely a precondition to conflict resolution or accommodation—is rendered impossible from the outset. Instead, the simplest criteria are used to detect differences, to search for means to reduce these differences, and finally to apply operators to "present objects" in order to transform them into "desired objects." It is, in fact, entirely reasonable, if "reason" means instrumental reason, to apply American military force, B-52's, napalm, and all the rest, to "communist-dominated" Viet Nam (clearly an "undesirable object"), as the "operator" to transform it into a "desirable object," namely, a country serving American interests.

The mechanization of reason and of language has consequences far beyond any envisioned by the problem solvers we have cited. Horkeimer, long before computers became a fetish and gave concrete form to the eclipse of reason, gave us the needed perspective:

> Justice, equality, happiness, tolerance, all the concepts that... were in preceding centuries supposed to be inherent in or sanctioned by reason, have lost their intellectual roots. They are still aims and ends, but there is no rational agency authorized to appraise and link them to an objective reality. Endorsed by venerable historical documents, they may still enjoy a certain prestige, and some are contained in the supreme law of the greatest countries. Nevertheless, they lack any confirmation by reason in its modern sense. Who can say that any one of these ideals is more closely related to truth than its opposite? According to the philosophy of the average modern intellectual, there is only one authority, namely, science, conceived as the classification of facts and the calculation of probabilities. The statement that justice and freedom are better in themselves than injustice and oppression is scientifically unverifiable and useless. It has come to sound as meaningless in itself as would the statement that red is more beautiful than blue, or that an egg is better than milk.

As we ourselves have also observed, the reification of complex systems that have no authors, about which we know only that they were somehow given us by science and that they speak with its authority, permits no questions of truth or justice to be asked.

I cannot tell why the spokesmen I have cited want the developments they forecast to become true. Some of them have told me that they work on them for the morally bankrupt reason that "If we don't do it, someone else will." They fear that evil people will develop superintelligent machines and use them to oppress mankind, and that the only defense against these enemy machines will be superintelligent machines controlled by us, that is, by well-intentioned people. Others reveal that they have abdicated their autonomy by appealing to the "principle" of technological inevitability. But, finally, all I can say with assurance is that these people are not stupid. All the rest is mystery.

The reification of authorless complex systems permits no questions of truth or justice to be asked.

It is only a little easier to understand why the public embraces their ideas with at least equanimity and sometimes even with enthusiasm. The rhetoric of the technological intelligentsia may be attractive because it appears to be an invitation to reason. It is that, indeed. But, as I have argued, it urges instrumental reasonings, not authentic human rationality. It advertises easy and "scientifically" endorsed answers to all conceivable problems. It exploits the myth of expertise. Here too the corruption of language plays an important role. The language of the artificial intelligentsia, of the behavior modifiers, and of the systems engineers is mystifying. People, things, events are "programmed," one speaks of "inputs" and "outputs," of feedback loops, variables, parameters, processes, and so on, until eventually all contact with concrete situations is abstracted away. Then only graphs, data sets, printouts are left. And only "we," the experts, can understand them. "We" do—even if only to have good public relations—show our concern for the social consequences of "our" acts and plans. Planning papers, such as the one I quoted, almost always have an opening paragraph that makes a passing reference to the destructive potential of our instruments. And "we" do write essays on the social implications of our gadgetry. But, as I have remarked elsewhere, these pieces turn out to be remarkably self-serving.

The structure of the typical essay on "The impact of computers on society" is as follows: First there is an "on the one hand" statement. It tells all the good things computers have already done for society and often even attempts to argue that the social order would already have collapsed were it not for the "computer revolution." This is usually followed by an "on the other hand" caution, which tells of certain problems the introduction of computers brings in its wake. The threat posed to individual privacy by large data banks and the danger of large-scale unemployment induced by industrial automation are usually mentioned. Finally, the glorious present and prospective achievements of the computer are applauded, while the dangers alluded to in the second part are shown to be capable of being alleviated by sophisticated technological fixes. The closing paragraph consists of a plea for generous societal support for more, and more large-scale, computer research and development. This is usually coupled to the more or less subtle assertion that only computer science, hence only the computer scientist, can guard the world against the admittedly hazardous fallout of applied computer technology.

Mankind has never been destroyed; but this time man is *able to destroy everything.*

The real message of such typical essays is therefore that the expert will take care of everything, even of the problems he himself creates. He needs more money. That always. But he reassures a public that does not want to know anyway.

And what is the technologist's answer to such charges as are here made?

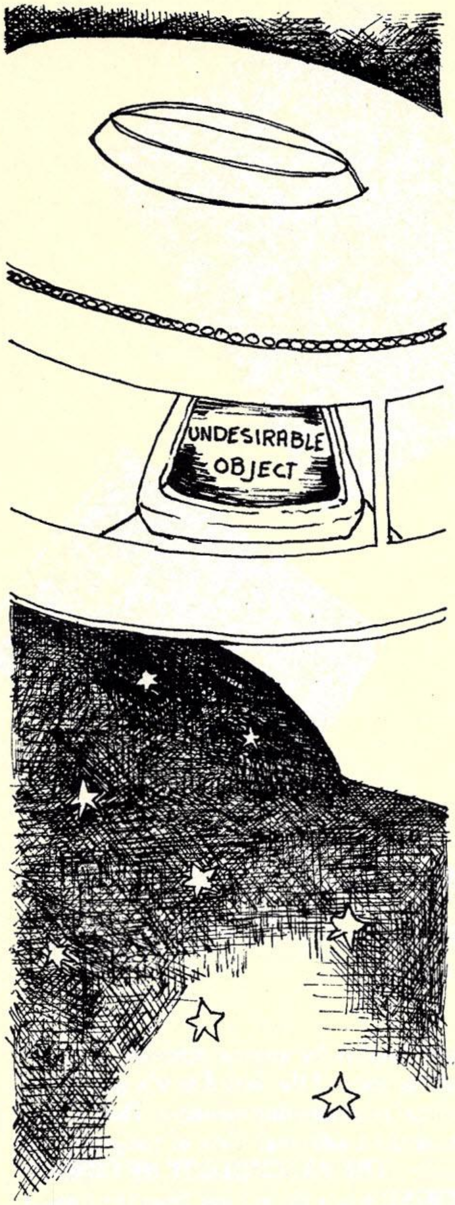
First of all, they are dismissed as being merely philosophical. For example, my paper on the impact of computers on society drew hundreds of letters, but only one from a member of the artificial-intelligence community. It came from a former student of Prof. Simon, and said in part,

> As far as society as a whole is concerned, the primary effects of computer technology are more important than their [sic] side effects. It is only the more philosophically inclined who find the potential side effects more important. . . . It takes a rare person to spend more than a few hours pondering the philosophical implications.

This is, of course, entirely consistent with Horkeimer's observation that language has lost even its right to speak in noninstrumental, that is, philosophical, terms. But a more directly significant answer was given by Dr. Kenneth B. Clark during a symposium held at M.I.T. not long ago. He had just expressed his distress that M.I.T. was not devoting more of its resources to the solution of social problems. He said (I quote from memory), "Here is a great institute devoted to and expert in science and technology. Why, in this time of anguish, do you not apply your instruments, your techniques, to the burning social questions of the times?"

I suggested that answers to the urgent questions of the time might not be found exclusively in science and technology. I said that his own search for technological solutions to great problems, for example, his proposal to give tranquilizers to world leaders on a regular basis, might be fundamentally misleading.

He responded, "I have very long ago come to the conclusion that answers to the great questions facing man at all



times can come only from rational thinking. The only alternative is the kind of mindlessness that, as we have seen, leads only to violence and destruction."

One might very well endorse that view—but only if by rationality something other than the mere application of science and technology is meant, if rationality is not automatically and by implication equated to computability and to logicality. The alternative to the kind of rationality that sees the solution to world problems in psychotechnology is not mindlessness. It is reason restored to human dignity, to authenticity, to self-esteem, and to individual autonomy.

Instrumental reason has made out of words a fetish surrounded by black magic. And only the magicians have the rights of the initiated. Only they can say what words mean. And they play with words and they deceive us. When Skinner contrasts science with common sense and claims the first to be much superior, he means his "behavioral science" and he means the "common" in "common sense" pejoratively. He does not mean a common sense informed by a shared cultural perspective, or a common sense that, for no "reason" at all, balks at the idea that freedom and dignity are absurd and outmoded concepts.

The technologist argues again and again that views such as those expressed here are anti-technological, anti-scientific, and finally anti-intellectual. He will try to construe all arguments against his megalomanic visions as being arguments for the abandonment of reason, rationality, science, and technology, and in favor of pure intuition, feeling, drug-induced mindlessness, and so on. In fact, I *am* arguing for rationality. But I argue that rationality may not be separated from intuition and feeling. I argue for the rational use of science and technology, not for its mystification, let alone its abandonment. I urge the introduction of ethical thought into science planning. I combat the imperialism of instrumental reason, not reason.

It is said that men could always be found who thought that their own time was filled with the greatest forebodings of catastrophes to come, and even that theirs were the worst of all possible times for the whole of mankind. Certainly, we who were alive and awake during the time fascism appeared to be almost everywhere victorious saw the grim reaper making ready for civilization itself. Somehow civilization survived that threat—a threat that today's youth can no longer comprehend. But it cannot be said that civilization survived it, or the Great War that preceded it by only two decades, wholly intact. We came to know as never before what man can do to his fellows. Germany implemented the "final solution" of its "Jewish Problem" as a textbook exercise in instrumental reasoning. Humanity briefly shuddered when it could no longer avert its gaze from what had happened, when the photographs taken by the killers themselves began to circulate, and when the pitiful survivors re-emerged into the light. But in the end, it made no difference. The same logic, the same cold and ruthless application of calculating reason, slaughtered at least as many people during the next twenty years as had fallen victim to the technicians of the thousand-year Reich. We have learned nothing. Civilization is as imperiled today as it was then.

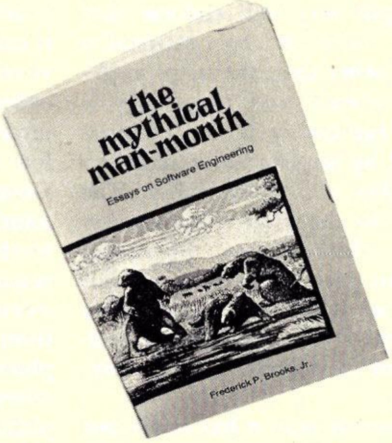
But if every time has heard the same Cassandra cry, then every time has also learned how little prophetic it seemed always to prove. Civilizations have been destroyed, many of them. But never mankind. But this time it is different. We are tired of hearing it, but we cannot deny it: this time man *is* able to destroy everything. Only his own decisions can save him.

It also used to be said that religion was the opiate of the people. I suppose that saying meant that the people were drugged with visions of the good life that would surely be theirs if they but patiently endured the earthly hell their masters made for them. On the other hand, it may be that religion was not addictive at all. Had it been, perhaps God would not have died and the new rationality would not have won out over grace. But instrumental reason, triumphant technique, and unbridled science *are* addictive. They create a concrete reality, a self-fulfilling nightmare. The optimistic technologists may yet be right: perhaps we have reached the point of no return. But why is the crew that has taken us this far cheering? Why do the passengers not look up from their games? Finally, now that we and no longer God are playing dice with the universe, how do we keep from coming up craps? ▼

Chip Encounters by Kork

The ROMshelf

For your further reading pleasure....


the mythical man-month
Essays on Software Engineering
Frederick P. Brooks, Jr.

TRAVELS IN COMPUTERLAND
OR, INCOMPATIBILITIES AND INTERFACES
Ben Ross Schneider, Jr.

THE PSYCHOLOGY OF COMPUTER PROGRAMMING
THE PSYCHOLOGY OF COMPUTER PROGRAMMING

TRAVELS IN COMPUTERLAND

Or, Incompatibilities and Interfaces

by Ben Ross Schneider, Jr.
Paperback, 244 pages
$6.50

Leave it to a stage-struck English professor (the London stage 1660-1800, at that) to come up with a rollicking outsider's view on the mad world of computers. A Canterbury Tale of all that went wrong—and some things that went right in a novice's attempt to set up a computer-accessible information base for scholars in theater, drama, and history. Must reading for anyone dealing with information retrieval or involved with computer-assisted education and research.

THE MYTHICAL MAN-MONTH

Essays on Software Engineering

by Frederick P. Brooks, Jr.
Paperback, 195 pages
$7.50

A collection of thought-provoking essays on the management of large-scale computer-programming projects, this volume is perhaps best introduced by listing some of the chapter titles. These include The Tar Pit; Aristocracy, Democracy, and System Design; Why Did the Tower of Babel Fail; Ten Pounds in a Five-Pound Sack; Plan to Throw One Away; and Hatching a Catastrophe. Ideal for reading on the train to work—though it might make you decide to switch occupations.

THE PSYCHOLOGY OF COMPUTER PROGRAMMING

by Gerald M. Weinberg
Hardcover, 286 pages
$9.95

This volume offers insights for anyone interested in software development, in spite of the fact that it's geared towards the professional programming manager. The reason? Unlike books taking the traditional view of programming as a machine activity, **THE PSYCHOLOGY OF COMPUTER PROGRAMMING** treats the subject from the human viewpoint—considering some current programming milieus, one might even say the humane viewpoint. Covers such interesting concepts as how the program reflects the writer's personality, how his desk and its accoutrements affect program development, and why aptitude tests really don't work in this field.

Please send coupon to:

The ROMshelf
Route 97
Hampton, CT 06247

☐My check is enclosed.
☐Master Charge #__________ Exp. date______
☐BankAmericard/Visa #__________ Exp. date______

Name__________
Address__________
City______ **State**______ **Zip**______

From the ROMshelf

Travels in Computerland $______
by Ben Ross Schneider, Jr.
The Mythical Man-Month $______
by Frederick P. Brooks, Jr.
The Psychology of Computer Programming $______
by Gerald M. Weinberg

Please add fifty cents per book for postage and handling.

Total $______

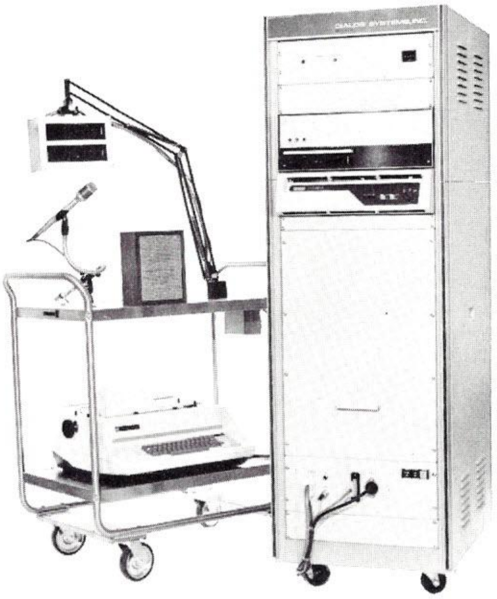
Please allow 4-6 weeks for delivery. Feel free to photocopy this page if you wish to keep your ROM intact.

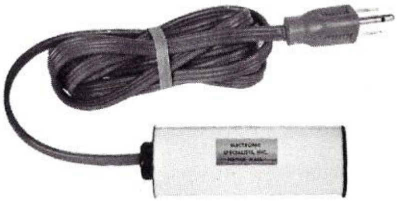
Helping Hand

Designed for immobilized hospital patients, the Dialog 117 System seems to have a great potential for home use as well. Direct voice control permits activation and deactivation of bedmotors, lights, typewriters, telephones, computer games, and various communication devices.

The system uses an electronic display for command verification and prompting. Converting sound waves into an amplified digital form understood by the system's word processor, Dialog 117 has 99 words at its command. Foreign accents and dialects are supposedly more comprehensible to the machine than to a nurse or other attendant.

Dialog Systems, Inc.
32 Locust Street
Belmont, Massachusetts 02178
(617) 489-2830

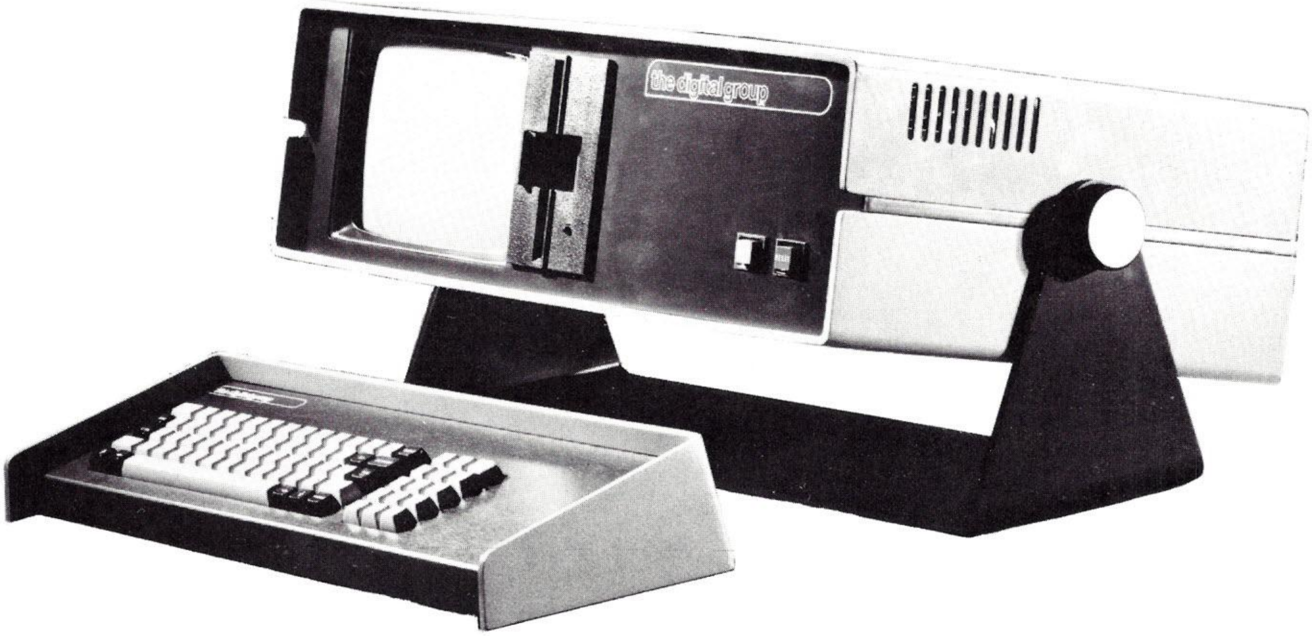




Shockproof

Worried about your micro being fried by lightning or other line surges? Does the power plant down the road dump on your peripherals? Worry no more. The Line-Cord Transient Suppressor will handle your wiry trespassers. Available in 2-prong plug/socket ($11.50) or 3-prong plug/socket ($14.50) from

Electronic Specialists
Box 122
Natick, Massachusetts 01760
(617) 655-1532



Mr. Bytemaster, I Presume

The Digital Group enters the out-of-the-box-and-ready-to-run computer fray with its spaceship-design Bytemaster. The top-of-the-line Master 4 model comes complete with Mini-Disk system and 32K of memory. You can add an extra 32K if you're in an expansive mood. All Bytemasters are geared to accept a range of peripherals, including monitors, printers, disk drives, and so on.

The Digital Group
P.O. Box 6528
Denver, Colorado 80206
(303) 777-7133

ROM

COMPUTER APPLICATIONS FOR LIVING

March/April 1978

This month's ROMfold features the mothership in Close Encounters of the Third Kind descending over Devil's Tower.

Photograph courtesy of Columbia Pictures
©1977 Columbia Pictures Industries, Inc.
All Rights Reserved.

Run On Micros Run On

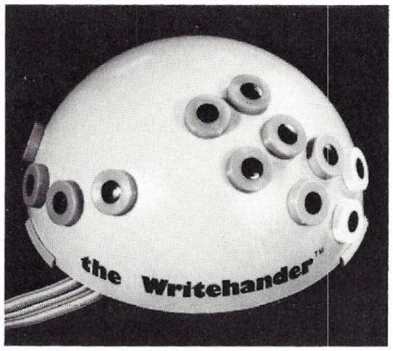
Type It!

If you have a problem justifying the outlay for a hard-copy printer, but you know your system will never be complete without it, consider the SELECTERM. A fully converted IBM Selectric II that functions as a regular typewriter as well as a printer, it can be connected directly to either parallel or serial ports. All logic is stored in the PROM and no additional software is needed.

Includes a special typing element with complete upper and lower alphanumeric ASCII characters. Functioning tab command, backspace, and vertical tab are standard. Dual pitch, correcting feature, and other specials may be ordered. Fully approved by Big Brother IBM, warranty is in effect, and service contracts are available. Complete at $1650 from

Micro Computer Devices
960 East Orangethorpe Avenue
Anaheim, California 92801
(714) 992-2270

Fastest Hand in the West



If they'd made it yellow, the Writehander would have looked like half a grapefruit. As it is, perhaps it can be best described as a bluish turtle with colorful Modern Danish pimples. Be that as it may, the Writehander could well be one of the most efficient data-entry devices to come down the I/O channel of technology in decades. (Of course someone will have to come out with a Lefthander as well, or there will be a lot of grousing.)

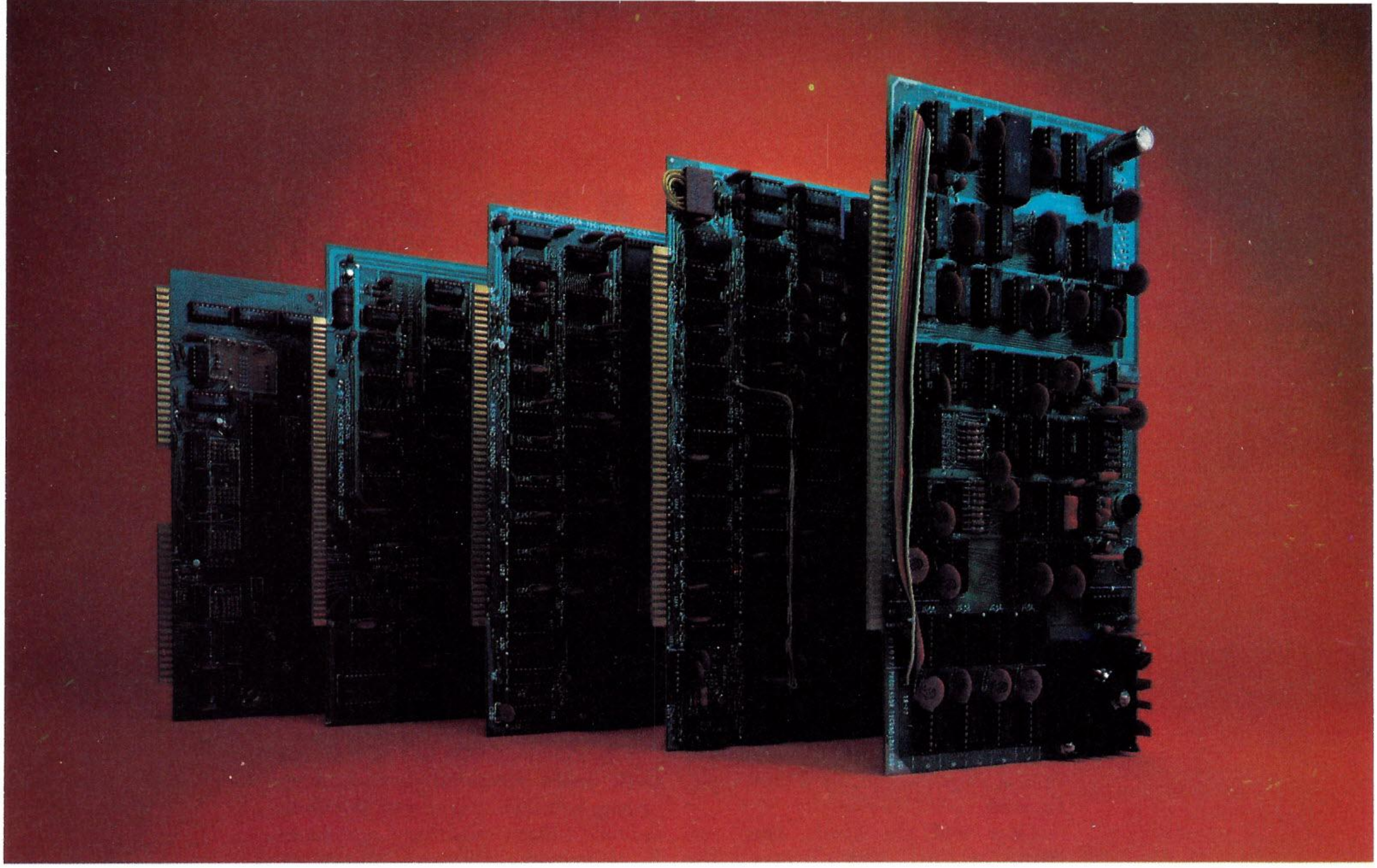
All 128 characters of the ASCII code can be handled on this one-handed keyboard. The advantage here, of course, is that it leaves the operator's other hand free to doodle, hang onto the telephone, or otherwise occupy itself. Looks like an "either you love it or you hate it" peripheral, but definitely worth looking into to check on how well it interfaces with your particular personality. Delivery from stock at $98 per hand from

NewO Company
246 Walter Hayes Drive
Palo Alto, California 94303
(415) 321-7979

Your Sol dealer has it.

AL: Birmingham: ICP, Computerland, 1550-D Montgomery Hwy., (205)979-0707. **AZ:** Tempe: Byte Shop, 1425 W. 12th Pl., (602)894-1129; Phoenix: Byte Shop, 12654 N. 28th, (602) 942-7300; Tucson: Byte Shop, 2612 E. Broadway, (602)327-4579. **CA:** Berkeley: Byte Shop, 1514 University, (415)845-6366; Costa Mesa: Computer Center, 1913 Harbor, (714) 646-0221; Hayward: Byte Shop, 1122 "B" St., (415)537-2983; Hayward: Computerland of Hayward, 22634 Foothill Blvd., (415)538-8080; Lawndale: Byte Shop, 16508 Hawthorne, (213)371-2421; Mt. View: Byte Shop, 1063 El Camino, (415)969-5464; Mt. View: Digital Deli, 80 W. El Camino, (415)961-2670; Orange: Computer Mart, 633-B W. Katella, (714) 633-1222; Pasadena: Byte Shop, 496 S. Lake, (213)684-3311; Sacramento: Micro-Computer Application Systems, 2322 Capitol, (916) 443-4944; San Francisco: Byte Shop, 321 Pacific, (415)421-8686; San Jose: Byte Shop, 2626 Union, (408)377-4685; San Rafael: Byte Shop, 509 Francisco, (415)457-9311; Tarzana: Byte Shop, 18424 Ventura, (213)343-3919; Walnut Creek: Byte Shop, 2989 N. Main, (415)933-6252. **CO:** Boulder: Byte Shop, 3101 Walnut, (303)449-6233; Denver: Byte Shop, E. 1st Ave. & University, (304)399-8995. **FL:** Ft. Lauderdale: Byte Shop, 1044 E. Oakland Pk., (305)561-2983; Miami: Byte Shop, 7825 Bird, (305)264-2983; Tampa: Microcomputer Systems, 144 So. Dale Mabry, (813)879-4301. **GA:** Atlanta: Computer Mart, 5091-B Buford, (404)455-0647. **IL:** Champaign: Computer Co., 318 N. Neil, (217) 359-5883; Numbers Racket, 623½ S. Wright, (217)352-5435; Evanston: itty bitty machine co, 1322 Chicago, (312)328-6800; Schaumburg: Data Domain, 1612 E. Algonquin, (312) 397-8700. **IN:** Bloomington: Data Domain, 406 S. College, (812)334-3607; Indianapolis: Data Domain, 7027 N. Michigan, (317)251-3139. **IA:** Davenport: Computer Store, 4128 Brady, (319)386-3330. **KS:** Overland Park: Personal Computer Center, 3819 W. 95th St., (913)649-5942. **MA:** Boston: Computer Warehouse Store, 584 Commonwealth, (617)261-2700. **MD:** Towson: Computer Etc., 13A Allegheny, (301)296-0520. **MI:** Ann Arbor: Computer Store, 310 E. Washington, (313) 995-7616; East Lansing: General Computer Store, 1310 Michigan, (517)351-3260; Troy: General Computer Store, 73 W. Long Lake Rd., (313) 689-8321. **MN:** Minneapolis: Computer Depot, 3515 W. 70th, (612)927-5601. **NJ:** Cherry Hill: Computer Emporium, 2438 Route 38, (609)667-7555; Hoboken: Computer Works, 20 Hudson Pl., (201)420-1644; Iselin: Computer Mart, 501 Rt. 27, (201)283-0600. **NY:** Endwell: The Computer Tree, 409 Hooper Rd., (607) 748-1223; New York: Computer Mart, 118 Madison, (212)686-7923; White Plains: Computer Corner, 200 Hamilton, (914)949-3282. **NC:** Raleigh: ROMs 'N' RAMs, Crabtree Valley Mall, (919)781-0003. **OH:** Columbus: Byte Shop, 2432 Chester, (614)486-7761; Dayton: Computer Mart, 2665 S. Dixie, (513)296-1248. **OR:** Beaverton: Byte Shop, 3482 SW Cedar Hills, (503)644-2686; Eugene: Real Oregon Computer Co., 205 W. 10th, (503)484-1040; Portland: Byte Shop, 2033 SW 4th Ave., (503) 223-3496. **RI:** Warwick: Computer Power, M24 Airport Mall, 1800 Post Rd., (401)738-4477. **SC:** Columbia: Byte Shop, 2018 Green, (803)771-7824. **TN:** Kingsport: Microproducts & Systems, 2307 E. Center, (615)245-8081. **TX:** Arlington: Computer Port, 926 N. Collins, (817)469-1502; Arlington: Micro Store, 312 W. Randol Mill Rd., (817)461-6081; Houston: Interactive Computers, 7646½ Dashwood, (713)772-5257; Lubbock: Neighborhood Computer Store, 4902-34th St., (806)797-1468; Richardson: Micro Store, 634 So. Central Expwy., (214)231-1096. **VA:** McLean: Computer Systems Store, 1984 Chain Bridge, (703) 821-8333; Virginia Beach: Home Computer Center, 2927 Va. Beach Blvd., (804)340-1977. **WA:** Bellevue: Byte Shop, 14701 NE 20th, (206)746-0651; Seattle: Retail Computer Store, 410 NE 72nd, (206)524-4101. **WI:** Madison: Computer Store, 1863 Monroe, (608)255-5552; Milwaukee: Computer Store, 6916 W. North, (414)259-9140. **D.C.:** Georgetown Computer Store, 3286 M St. NW, (202)362-2127. **CANADA:** Toronto, Ont: Computer Mart, 1543 Bayview, (416) 484-9708; First Canadian Computer Store, 44 Eglinton Ave. W., (416) 482-8080; Vancouver, B.C.: Basic Computer Group, 1438 W. 8th, (604)736-7474; Pacific Computer Store, 4509 Rupert, (604)438-3282.

Processor Technology



Subsystem B

Each board is a standout.
Together, they're a powerhouse.

In the beginning there were boards, thousands of them.

That's how we started in the business. Making memories and interfaces for other people's computers...and making them better.

Now that our own Sol has become the number one small computer, you might think we're putting less emphasis on our board business.

Not so.

We're just doing more creative things with them.

One neat package gets your computer on the air.

For example, we've built Subsystem B, which ties together five Processor Technology modules into a completely integrated system that makes other S-100 Bus computers work almost as well as our Sol.

Subsystem B includes a memory module, three input/output modules, a general purpose memory, and appropriate software.

A specialized software program called CUTER knits together your computer and its peripherals to create an integrated, smoothly working system.

It's the fastest, cleanest way to get on line, and it costs less than if you bought each module separately.

You get your choice of two low power, reliable memory modules in 8K or 16K capacity.

Our VDM-1 video display module (still $199 in kit) is the industry standard display device with over 6,000 in use.

Our CUTS high speed, low cost ($149 in kit) audio cassette interface is the most reliable on the market and is supported by our broad line of cassette software including Extended BASIC, FORTRAN,* PILOT,* FOCAL and numerous others.

And our 3P+S input/output module offers a low cost way to handle virtually all the I/O needs of any S-100 Bus compatible computer system. There are close to 10,000 in the field. Price is just $149 in kit.

Yes, we may have become the maker of the Number 1 small computer— the Sol. But we haven't neglected the quality of our board business. We can't afford to... because we use many of them in our own computers.

For our most recent literature and price list see your dealer or write Processor Technology Corporation, Box O, 7100 Johnson Industrial Drive, Pleasanton, CA 94566. (415) 829-2600.

Processor Technology

*Available soon.

U.S. prices only.

TIME
Real & Unreal

by Sandra Erikson

"Excuse me, do you have the time?"

How often have we asked or been asked that question? And how automatic is our glance towards the wrist, or the clock radio, or the kitchen dial? The delineation of time is all around us, as is time itself. Or is it?

Time and the clock. Our own "sense" of time—which caused Colin Pittendrigh to ask of man, "*Has* it a clock? Or *is* it a clock?"—is, along with its bounded self, duration, probably the most ambiguous of the perceived universe's variables. It is also—though some would philosophically deny the very existence of time—the variable whose measurement has achieved the greatest precision.

In the measurement of time, the brief second, rather than the hour, is the standard unit. Yet the second in today's technological society is of very long duration indeed. Milliseconds and microseconds, nanoseconds and picoseconds are second nature to the computer. Who knows when even femtoseconds (the million-millionth parts of a second) will creep into existence as a familiar concept? The computer, like man, cannot function without an internal clock, cannot relate to the real world without measuring the passing of time. Of course, there's time, and then there's time, not to mention real time.

While the concept of real time is bandied about the computer world with equanimity, no consistent definition for it has been agreed upon. One can at best describe some of the features of a real-time system or set an example in opposition to "the other kind" of system.

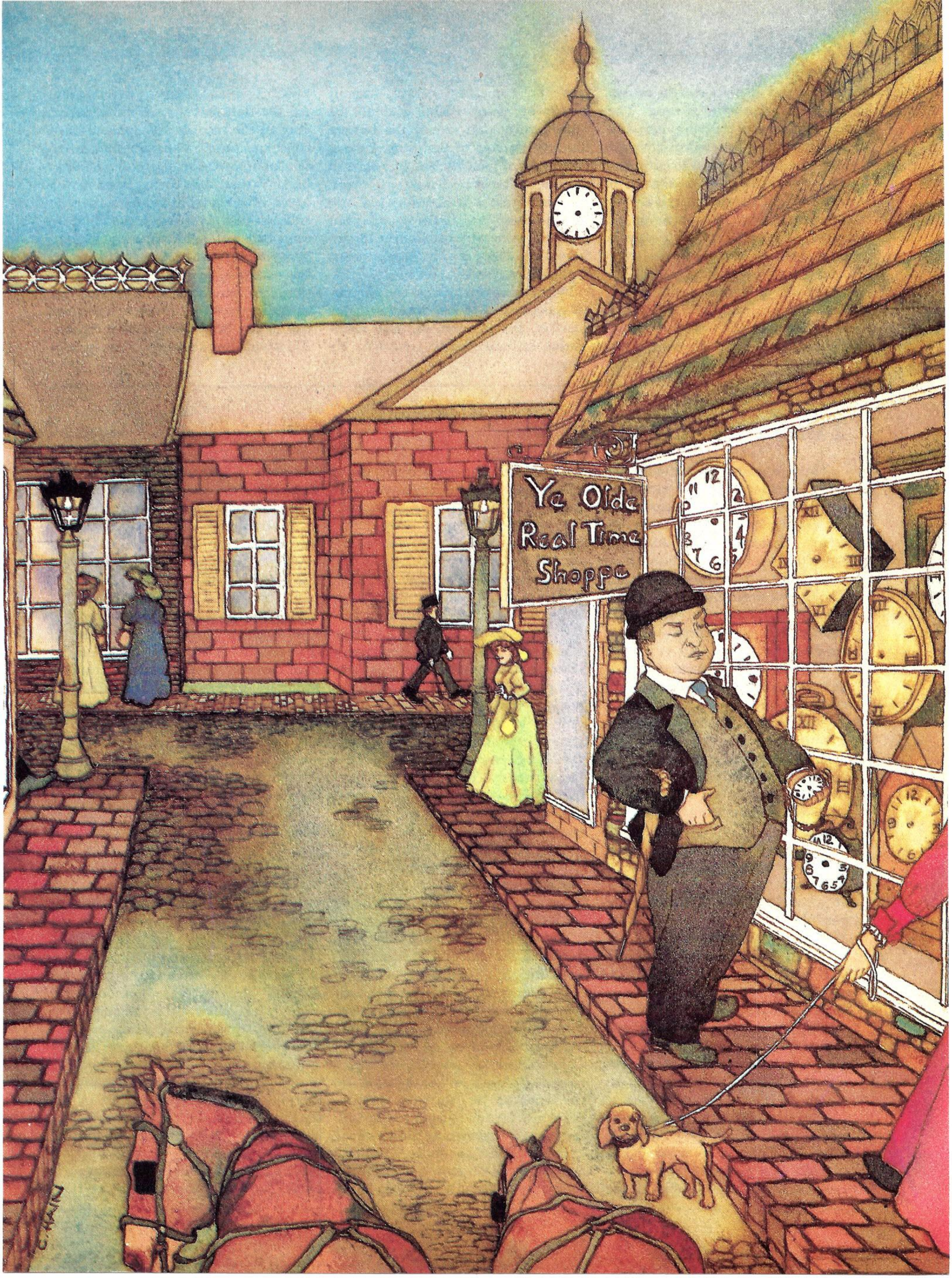
Back in the old P.C. (Pre-Computer) days, you walked into the bank and asked what your balance was. Assuming the teller knew you and wasn't out on a coffee break, he consulted the balance sheet and gave you the figure.

When the computer first arrived, the clerk would check the previous day's closing printout to give you the same balance (which unfailingly seemed to be not enough). Of course, this didn't indicate which transactions had cleared and which ones hadn't. The bank had thousands of accounts, and merely listing the totals for each one consumed considerable computer time, not to mention miles of printouts. For the most part, the time was wasted as well. After all, how many customers on a given day inquired as to their balance? A fraction of a percent of them at most. The problem was, the bank never knew who might ask, and so each account was totaled each day. After all, what would happen if your friendly banker answered your question concerning the balance with "I don't know"?

Of course, having the various accounts balanced—with the totals kept in the computer memory and only the specific ones required printed out as needed—would have been sufficient. That was the next step. Since, with this system, the printers were much less in demand, more data could be retrieved in less time. If one requested a balance, for instance, a further request for the amounts of checks cleared the previous day posed no problem. Fewer printouts—more information.

When information to be processed is arranged for the convenience of the computer—that is, all required data is collected, fed into the machine,

Illustrated by Cindy Hain


Ye Olde
Real Time
Shoppe
C. HAIN

crunched, and spewed out—a system called batch processing is used. The data is arranged for the computer, and the computer is instructed in advance what to do with it and what to output—*whether it's all needed or not.* In a real-time system, the computer is always in there, collecting data whenever it's available, and it delivers the processed information whenever it's needed.

Consider another example: the Glitchet Microworks payroll. Joe Glitchet, Jr., punches in on his card every morning and out again every evening. At the end of the week, his card is fed into the computer, which happily knocks off taxes, FICA, and all the other figures that reduce the bottom line to a shadow of the gross. Having taken care of Joe Glitchet, Jr.'s card, it goes on to one of his fellow employees.

It isn't quite as straightforward, nor as time-consuming, as it sounds, but that's the general idea. Collect the information. Once it's all there, process it chunk by chunk.

In real-time operations, the computer is hooked up to the clock itself. Joe Glitchet, Jr., punches in. The information is immediately recorded. He punches out and the same step is taken. All the data is collected as it occurs and all the records are updated on a continual basis. The computer is extended into the real world to collect data, rather than being spoonfed a batch of precompiled information. It collects this data by responding to some external event like a rise in temperature, an increment of light, or, in this example, Joe Glitchet putting his time card into the clock. This response to the entry of information has given rise to the term *event-driven system.* By implication, then, real-time systems

Time and Time Again

Time is about as definable as God. Whether it flows or passes us, or whether we advance through it, all but the process—or rather, its results—is unintelligible. After all, how do you measure the number of seconds that pass by per second? The flow of time would have to be a change in respect to something else—call it hypertime. But since hypertime itself must flow, or we flow through it, then hypertime must be held against hyper-hypertime for measurement.

Consider man or his consciousness flowing through time/space. Do future events pop into existence as man reaches "now"—or are they there all along? And can time be represented in the form of a change of itself?

There are philosophers who doubt what may well be the safest view—that time is an illusion. After all, changing the future is no more possible than changing the past. When a person reaches a fork in the road and decides to turn left rather than right, well, the turn he takes *is* what the future was. Left. It was never right. This is not a predeterministic view; one can't apply laws and deduce the future from the past, for there may well be futures not even connected, especially in an organized fashion, to earlier ones.

The so-called process philosophers such as Alfred North Whitehead and Henri Bergson, on the other hand, consider time to be a real and very essential metaphysical fact. The catch? Well, the concept of time can be grasped only intuitively. It's there and you accept it in an almost Kierkegaardian leap of faith.

The very attempt to define time cannot be infused with the precise verbal distillation granted the delineation of, say, horse. Rather, a definition of time evolves with the acquisition of verbal skills at a level where words such as *now, tomorrow, later,* and *hour* can be tied together in meaningful thought. In our modern scientific culture, it was only to be expected that this progressive, shifting, nebulous *time* should be systematized as a quantitative process. The quality of time may still be questioned, but there's something there, by gum, and it can be measured. Accurately.

As it turns out, there are two independent fundamentals of time, one with which a computer would be more at home and one which appeals to the human senses.

Dynamic time is observable. It has been the cultural mainstay of most societies since, shall we say, time immemorial. Dependent on the motion of bodies, it has as its most obvious example the orbital patterns of planets and satellites. Called *ephemeris time,* it was most commonly based on the rotation of the moon or the sun, with various corrective factors to compensate for the fact that everything didn't work out the way it was supposed to.

Dynamic time took a giant step forward with the discovery by Galileo, around 1583, that the oscillation period of a pendulum is, to all intents and purposes, independent of the amplitude of its swing. Here was a dynamic force that could readily be harnessed to a mechanical display movement to measure time. By 1656 a Dutch mathematician, Christiaan Huygens, had put it all together and constructed the first pendulum clock. A little less than three hundred years later, the dynamic measurement of time reached a constant of 0.001 second per day. At that point the pendulum clock could not be improved upon, and progress stood still. Dynamic-time measurement had reached its peak.

The advent of the electronic era, however, opened a new way to delineate time. Electromagnetic time, though not observable like dynamic time, is also dependent in a sense on motion. Here it is the cycles of electromagnetic radiation which are enumerated. This is the principle behind today's superaccurate atomic clocks. The key difference is that dynamic time involves the motion of material bodies; electromagnetic time, the time of the computer, does not. But perhaps that brings us, as humans, closer to the innate time of the universe. Norbert Wiener speculated that the human sense of time depends on the electrical oscillation of the alpha rhythm in the brain. Man the living clock is real time in the universe—yesterday, today, and maybe tomorrow.

must be running all the time, or at least all of the time during which a recordable event might occur.

Simultaneously, there are specific time frames to be dealt with. Certain tasks must be completed in the time alloted them. Joe Glitchet isn't about to hang around the time clock waiting for his card to be read. In it goes. Pling. Out it's pulled. That's that. If the computer hasn't grabbed the data, it's lost. The specific time frame alloted for the collection of specific data is known as *response time.* Measured in microseconds, or at most seconds, response time is essentially what real-time computers are all about.

It's the quality of responsiveness that sets real-time systems apart from previous modes of computer operation. How responsive a system must be before it's considered a real-time one is a relative question. All but the largest real-time operations require hardware and software that provide a response time of within a second. In really large systems, of which an airline's seat-reservation system is a commonly observed example, three to five seconds is more the norm. Adhering to this ideal response time, however, can be prohibitively expensive. Consider, for instance, the almost zero probability of all reservation clerks at every terminal making a booking at the same time. If the system were geared to handle this capacity with a response time of five seconds or less, it would be vastly underutilized most of the time.

On a more personal level, the interactive home computer can and really must keep its response level close to the under-a-second ideal. In the case of a time-sharing system, for instance, a response time of thirty seconds could defeat the whole purpose of the setup. The same holds true for home control.

If your computer is monitoring the temperature and humidity in the greenhouse, checking the respiration rate of the child upstairs, and printing out the revised inventory of the larder—and you have to wait a minute before your micro makes its move in Othello—well, chances are you're better off getting small dedicated units for control. That way you can save your "big" computer for more personal interactive use.

The key to the future of real-time systems lies in the fact that some operations become possible only by virtue of rapid computer controls. Although these do not apply to the home environment, their application affects us all, and there may possibly be some spinoffs on the personal level. Today's fast-flying fighter planes would be relatively inoperative without radar-jumping capabilities—the equipment for which is real time based in the microsecond response realm. The same type of equipment is being pioneered as a means of radar-responsive automobile control. James Bond's car is certainly decades away, but at least some serious thought is being given to real-time computer-controlled personal transportation.

Perhaps the most familiar real-time system, and the one furthest along in its development, is the point-of-sale terminal. This terminal captures all the information concerning the financial inventory aspect of your purchase. Taking it one step—it's a huge step, involving some yet-to-come computer architecture that puts processing beyond the speed of light—further, and there's no reason why the point-of-sale terminal, the bank's computer, and your home computer should not eventually be tied together in one massive real-time network.

You buy a dozen eggs and a six-pack at the corner deli (if it's still in business). The point-of-sale terminal deducts the purchase from the store's inventory, the bank's computer deducts the money from your bank account, the government collects its sales tax, and your home computer scratches beer off the shopping list. It didn't scratch the eggs off because they weren't on the list in the first place. Oh well, no one's perfect, even in real time—and that's a long time away. ▼

Just a Second

So firmly established are the units of time that, although decimalization is being advocated in almost all measurements, horology remains unscathed except for a gradual switch from the once standard sweeping hands to digital display. True, the second has been decimalized into milliseconds (0.001 sec.), microseconds (0.000 001 sec.), and picoseconds (0.000 000 001 sec.). But the standard unit, the second itself, bears no decimal relation to its larger cousins. The ratios 60 : 1 and, on a larger scale, 24 : 1 have dominated man's division of time for most of history.

The ratios developed some five thousand years ago or more, in that heartland of civilization known as Mesopotamia. Perhaps it was the wide horizons on the lowlands of the Tigris and Euphrates valleys, perhaps the development of a civilization freed by the fortuitous circumstances of the Fertile Crescent from a day-to-day struggle for survival—whatever the case, sky watching became part of life. The regularity of movement in the stellar kingdom was noticed, and the march of heavenly bodies across the sky was embodied into measurable units and clocks. Angles formed between the earth and the stellar objects above were measured. Somewhere along its development, Mesopotamian culture (although its inhabitants had no more fingers than we do today) adopted a number system based on 60. There were 360 days in a year, 30 days in a month. Out of it all came the sexagesimal system—which, incidentally, has a mathematical advantage. The decimal system's dominant 10 is evenly divisible only by 2 and 5. Base 60, on the other hand, can be evenly divided by 2, 3, 4, 5, 6, 10, 12, 15, 20, and 30. Great if you like division. And the Mesopotamians did.

Among other things they came up with the 360-degree circle (60 conveniently multiplied by 6), which in turn gave a right angle of 90 degrees and all those other standard angles and arcs like 30 degrees and 60 degrees and so on.

But single degree was still rather coarse, even for the tools available in those days. So what could be more natural than to split it into fractions. Divide by 60, of course. And what do you get? A minute of angle—or time. Divide it again and there's a second of angle—or time. The hands of a clock sweep across its face mimicking the rotation of the heavenly bodies around earth. And time marches on.

The Absolute Time Clock

For computer applications in which the machine relates closely to people, it's nice to have the computer know what time it is. Until the little beast has learned to tell time, you will have to keep feeding it things to do and data to work on. When it's through with each task, it will sit there and stare at you, waiting for its next feeding. You can't send it away to work on a low-priority, tedious problem and expect it to check back in ten minutes, or at 3:30 P.M.

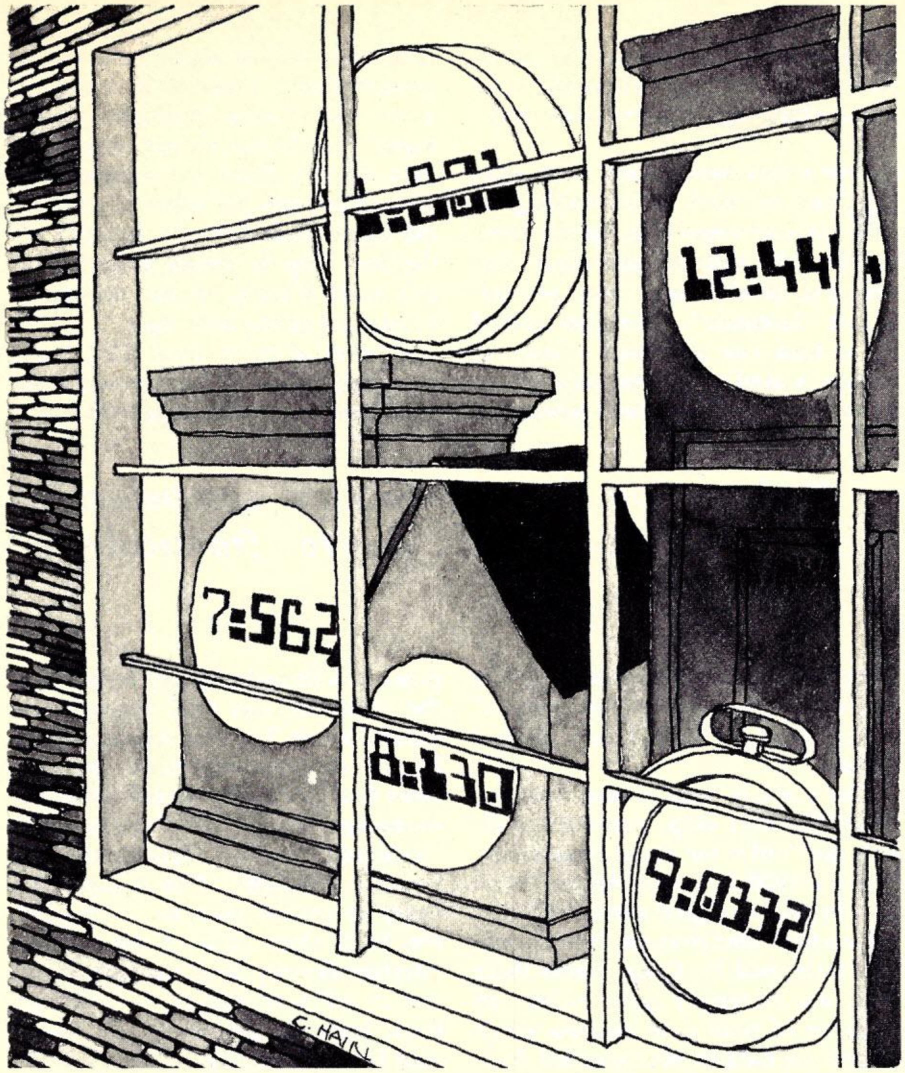
No doubt about it, computers have a very poor sense of time. Perhaps you thought that because computers have clocks within them, they must be frantically aware of the time. Alas, the word *clock* has been stretched here somewhat. In a computer, it refers to a continuous signal used to determine the sequence of events going on inside. It has no necessary relationship to the time of day.

However, the clock signal can be used by the programmer to generate simple delays. The time taken by each clock cycle is specified by the computer manufacturer (500 nanoseconds is common for most 8080 microprocessors, for instance), and the number of clock cycles required for each instruction is published by the microprocessor chip manufacturer (8080s require from three to eighteen clock cycles per instruction). By looking up the cycle counts for a list of instructions and multiplying by the clock-cycle time, a programmer can get a good idea of how much time the list of instructions will occupy. The programmer can build in "timing loops" in which the computer executes a delay by running through a set number of meaningless instructions before proceeding. This is the way in which most microcomputers working with video displays slow down or speed up their rate of display according to the user's preference.

Note that we haven't called the clock a real-time *clock.*

If the program is simple and of great regularity, such "cycle counting" techniques can be expanded to yield the time of day. The programmer has to provide for a number, stored somewhere in the computer's memory, designating the time of day. Periodically, the computer program will update this number by adding in the time since the previous update. It works only if the program is predictable—high-level programs, especially interactive programs like BASIC, vary their sequences of operation wildly depending on what the user is doing at the moment.

There are peripheral devices such as disks which are so rude as to stop the op-



by
Lee
Felsenstein

eration of the CPU altogether while they meddle with the contents of memory (direct memory access, or DMA). When the computer wakes up again, it has no idea how much time has passed.

If we want our computer to know that it's lunchtime and we aren't to be disturbed until 1:12 P.M., we have to give the computer a clock to which it can refer for the correct time—a digital clock, of course, built from the same integrated electronic circuits as those used for digital clocks in the home. Since these are electronic devices very much like the ones in the computer, it is not a complex task to hook up a clock that the computer can read. One such "absolute-time clock" can be built from parts readily available from mail-order sources.

No doubt about it, computers have a very poor sense of time.

Note that I haven't called the clock a *real-time clock,* a term often used incorrectly. A real-time clock is, strictly speaking, a timer that the computer sets and that "goes off" some fixed time later. The computer can ignore it or reset it. It can be used to tell the time of day, but that takes some additional software or hardware to maintain a count. In the case of the clock described here, the digital-clock chip maintains this count and should not be referred to as a real-time clock.

This clock circuit is not specific to any one computer, but is intended to connect to the computer through an eight-bit parallel interface. The computer must feed at least one bit to the clock (telling it to present the next digit for examination); the clock pre-

ROMtutorial ROMtutorial

Peripherals: Devices outside the computer which are controlled by the computer. Printers, teletypes, and tape drives are some examples.

Disk: A means of storing information on what amounts to a magnetic phonograph record. Unlike a phonograph record, a disk storage device can position the read/write head at any track without having to wait for it to "play through."

Central processing unit (CPU): The "thinking" portion of a computer. It decides where to move information, what to do with it once it's there, and where it should look for its next instructions.

Microprocessor: The "thinking" section of a computer is called the central processing unit (CPU), or simply the processor. If it's so small that you need a microscope to examine it, it's called a microprocessor.

Diagrams by the author,
halftone by Cindy Hain

ROMtutorial ROMtutorial

Loops: Segments of a program which are executed a number of times. Usually they consist of a series of instructions followed by a test to see whether the program should execute the preceding instructions, or continue with the instructions that follow.

Higher-level language: Programming language oriented toward the problem to be solved or the procedures to be used. Contrasts with machine language, which is the basic language of a computer. Programs written in machine language require no further interpretation by a computer. Usually, however, a higher-level language is easier to program in than a lower-level language.

BASIC: Acronym for Beginner's All-purpose Symbolic Instruction Code. An easy-to-learn, easy-to-use programming language especially adapted for use with mini- or microcomputers, as well as time-sharing systems. It provides anyone using the computer with instantaneous feedback on whether he's doing all right or making a mistake.

Bit: The simplest unit of information in computing—the condition of being either on or off. Electrical circuits are well-suited to dealing in bits, since it's no problem (usually) to tell the difference between a high and a low voltage.

Parallel interface: A device that accepts data when all bits of a particular character are being transmitted simultaneously.

I/O: Input/Output. The electrical channels through which a computer moves information to and from the outside world.

Binary-coded decimal: Ordinary decimal numbers aren't represented as decimal numbers in a computer. Instead, they must be coded in binary. Binary-coded decimal (BCD) is a way of doing this, representing each decimal digit (0-9) as a binary integer four bits long (0000-1001).

TTL: Transistor-transistor logic. One of several kinds of electronic circuits which can be hooked into digital integrated circuits. Each kind has its advantages and weaknesses. TTL is the dominant digital circuit used today.

Buffer: A temporary storage area which is used to equalize or balance different operating speeds. For example, a buffer can be used between a slow input device, such as a typewriter, and the main computer which operates at a very high speed.

sents back the digit (four bits), along with a three-bit code telling the computer which digit is being presented.

Various construction techniques can be used in assembling an absolute-time clock. For the home builder, the simplest technique that will give good results uses printed-circuit "laminate": fiberglass coated on at least one side with a layer of copper, scored with a hacksaw to separate areas used for various power and ground signals. This "planar" structure does a lot to reduce the pesky "glitches" which plague high-speed digital circuits.

> *Perhaps you thought because computers have clocks inside, they're frantically aware of the time.*

The clock is built around the National MM5311 chip, currently available mail order for about five dollars. This is a MOS LSI chip designed to be used in plug-in digital clocks with very few external components. It has a feature little used in digital clocks these days, but extremely useful to us—an extra set of time code outputs which deliver the data in binary-coded decimal (BCD).

Most clocks use a seven-segment bar display. This chip supplies that information as well, with one output pin for each bar so that the proper segment combination lights up for each number. But we don't need that.

The chip also "multiplexes" the six digits of time information by providing a set of six outputs, only one of which is turned on at a time. Each "digit select" output thereby notifies the external world which of the seconds, minutes, or hours digits is being presented. This information is presented to the parallel interface along with the four-bit binary code for the digit's value (0 through 9). The computer can accept this information, read in the digit, and put it in its proper place in the time count, which it maintains somewhere in memory.

The multiplexing concept was intended to allow six digits of display, each having seven segments, to be connected to the chip without requiring forty-two separate connections. Inside the chip is a counter circuit which steps through a sequence that determines which digit's data will be presented next. This counter is stepped under control of a "multiplex timing input" pin. In our circuit, one bit of data coming from the computer is used to tell this pin when to advance the count. Unfortunately, we can't tell the clock chip to present a specific digit's worth of data. We can only tell it to advance to the next digit.

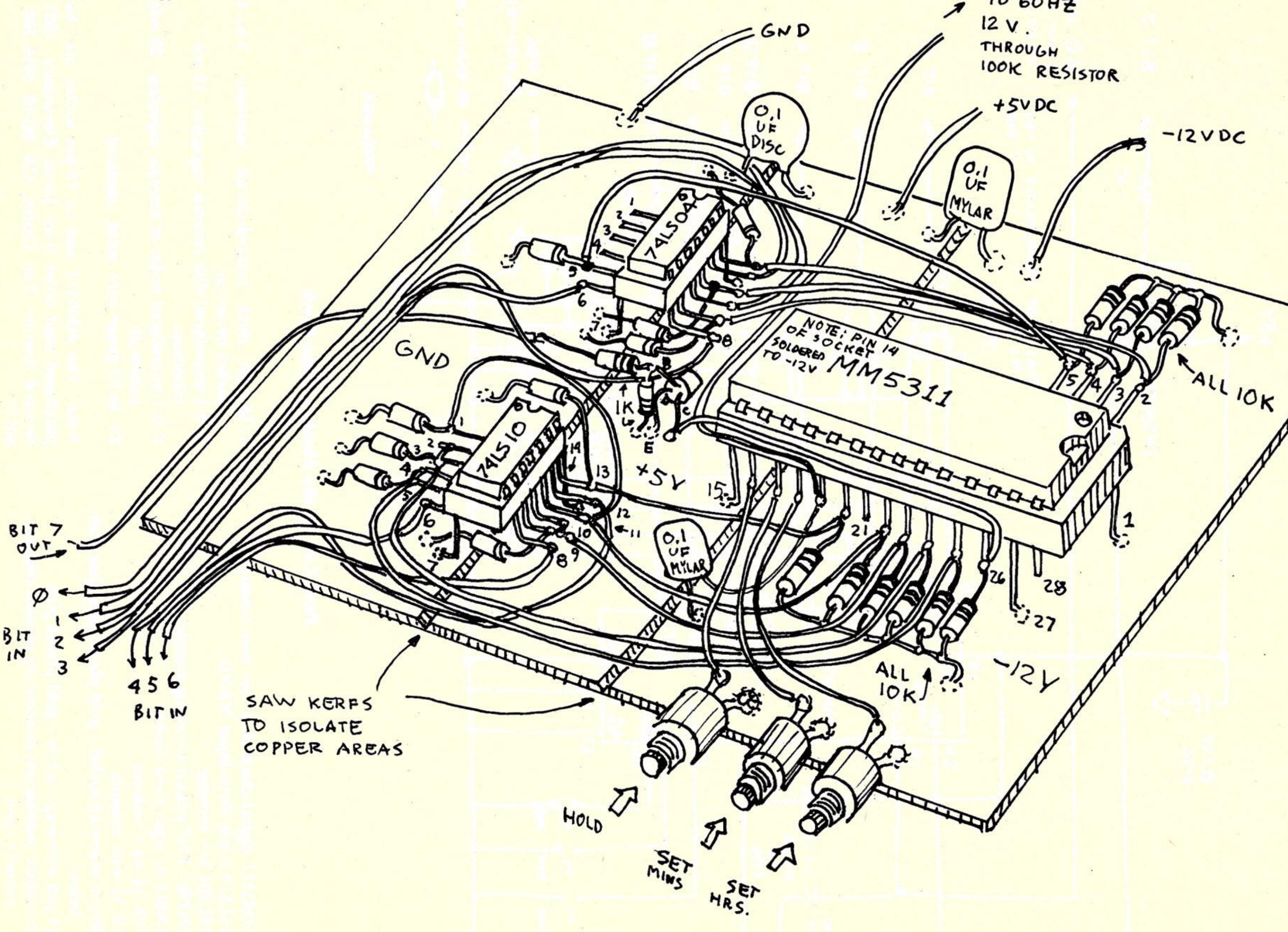
A series of three-input TTL gates (74LS10) converts the one-out-of-six code presented by the digit enable outputs into a three-bit binary code which the computer can read through the parallel input. The time data is already binary coded, so the circuit presents those four bits to the parallel interface unchanged except for inverting and "buffering" each bit through the hex inverter (74LS04). This is done because we cannot hook the outputs of the MOS clock chip to the parallel interface directly, since the data comes out at different voltage and current levels from the standard TTL levels used by the interface.

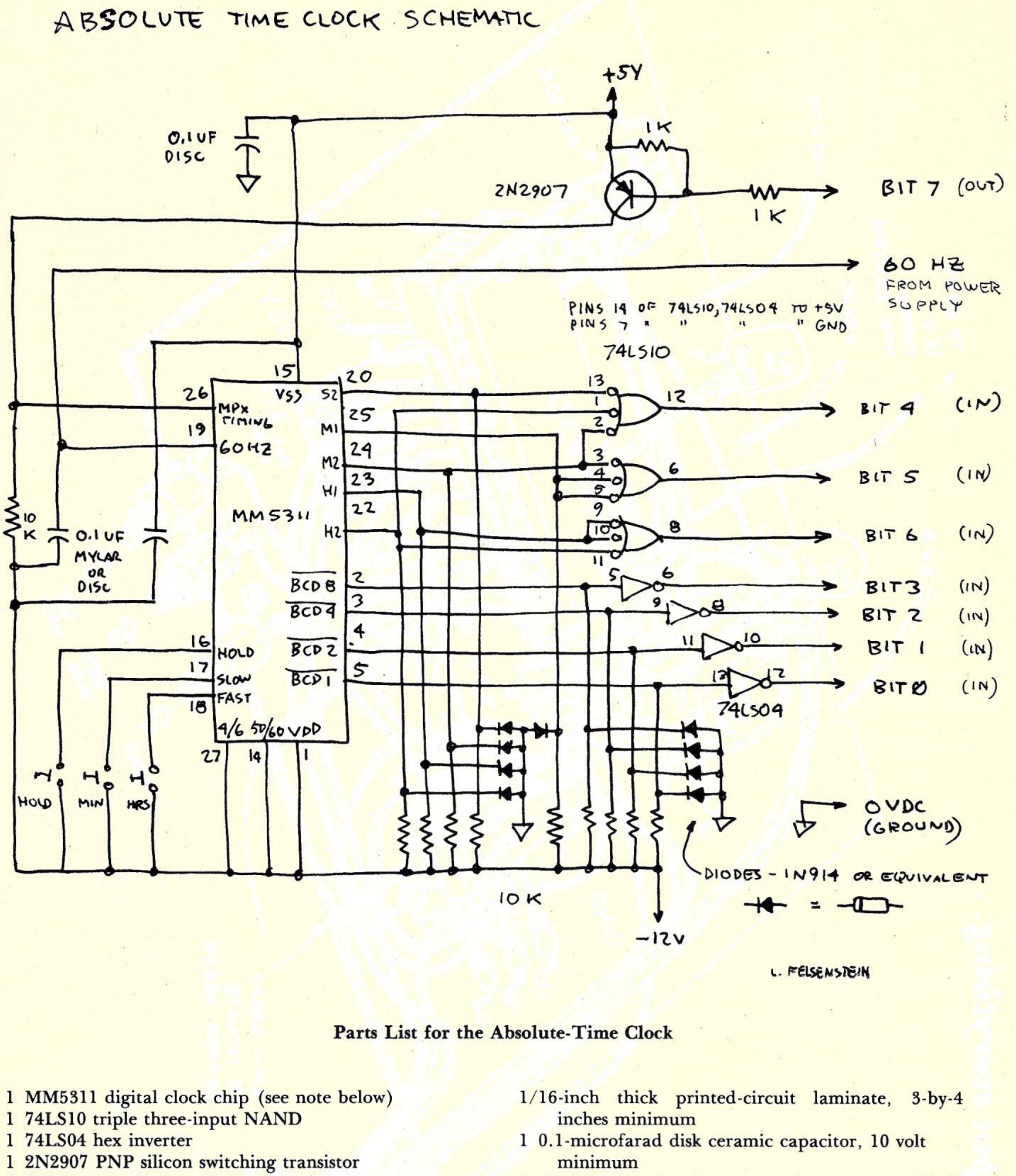
Power for the clock chip and the TTL gates comes from a +5-volt and −12-volt regulated supply capable of supplying about 50 milliamperes. The 60-hertz "time base signal" comes from the output of the power transformer through a resistor. A capacitor is added to smooth out spikes and other unwanted noise from the a.c. power line, as they can erroneously advance the clock's counters.

The power supply should be built separately so that it can be left on when the computer is turned off, thus eliminating the necessity of setting the clock every time you power up. Three pushbuttons are provided for setting the time: *Hold, Slow Set,* and *Fast Set.* To set the time, hold down both the *Fast* and *Slow Set* buttons until the hour is correct. Next, hold down only the *Slow Set* until the next minute is displayed. Then press the *Hold* button until the indicated time occurs. Upon release of *Hold,* the count will begin at the start of the second.

Presto! Your computer will now have time on its hands. ▼

Where to put everything. . .

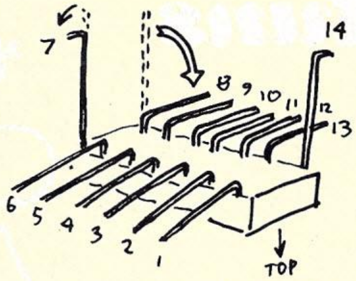

GND
TO 60HZ
12 V.
THROUGH
100K RESISTOR
+5VDC
-12VDC
0.1 UF DISC
0.1 UF MYLAR
74LS04
NOTE: PIN 14 OF SOCKET SOLDERED TO -12V
MM5311
5 4 3 2
ALL 10K
GND
1K
B
C
E
74LS10
+5V
1 2 3 4 5 6 7 8 9 10 11 12 13 14
15
21
26
28
27
1
0.1 UF MYLAR
-12V
ALL 10K
BIT 7 OUT
0
1
2
3
BIT IN
4 5 6
BIT IN
SAW KERFS TO ISOLATE COPPER AREAS
HOLD
SET MINS
SET HRS.

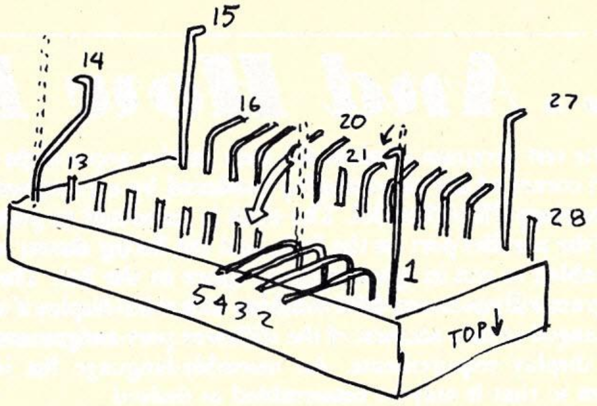

ABSOLUTE TIME CLOCK SCHEMATIC
+5V
0.1 UF DISC
1K
2N2907
1K
BIT 7 (OUT)
60 HZ FROM POWER SUPPLY
PINS 14 OF 74LS10, 74LS04 TO +5V
PINS 7 " " " " GND
74LS10
15
VSS
S2
20
26
MPX TIMING
25
M1
19
60HZ
M2
24
23
H1
22
MM5311
H2
10 K
0.1 UF MYLAR OR DISC
13
1
2
12
BIT 4 (IN)
3
4
5
6
BIT 5 (IN)
9
10
11
8
BIT 6 (IN)
2
BCD 8
5
6
BIT 3 (IN)
3
BCD 4
9
8
BIT 2 (IN)
4
BCD 2
11
10
BIT 1 (IN)
16
HOLD
17
SLOW
5
BCD 1
13
12
BIT 0 (IN)
18
FAST
74LS04
4/6 50/60 VDD
27
14
1
HOLD
MIN
HRS
0VDC (GROUND)
DIODES - 1N914 OR EQUIVALENT
10 K
-12V
L. FELSENSTEIN

Parts List for the Absolute-Time Clock

- 1 MM5311 digital clock chip (see note below)
- 1 74LS10 triple three-input NAND
- 1 74LS04 hex inverter
- 1 2N2907 PNP silicon switching transistor
- 9 1N914 or equivalent diodes
- 9 10K ¼-watt resistors
- 2 1K ¼-watt resistors
- 1 28-pin wirewrap socket—gold plated pins (see note below)
- 2 14-pin wirewrap sockets—gold plated pins
- 3 miniature pushbutton momentary contact switches, normally open
- 1/16-inch thick printed-circuit laminate, 3-by-4 inches minimum
- 1 0.1-microfarad disk ceramic capacitor, 10 volt minimum
- 2 0.1-microfarad mylar or ceramic capacitor, 25 volt minimum
- 24- to 30-gauge wire, solid, insulated

Note: The MM5311 and its 28-pin socket may be purchased mail order from James Electronics, 1021 Howard Avenue, San Carlos, CA 94070 (415-592-8097).


7
14
8
9
10
11
12
13
6
5
4
3
2
1
TOP


15
14
16
20
27
13
21
28
1
5432
TOP

14-PIN SOCKET PREPARATION:

BEND 1/16" OF END OF PINS 7 AND 14 OVER 90°

BEND PINS 1 – 6, 8 – 13 90° OUT 1/16" FROM SOCKET BODY.

SOLDER CATHODE (BANDED) END OF DIODES TO PINS 1, 3, 5, 9 AND 13 OF 74LS10 SOCKET AND TO PINS 5, 9, 11, 13 OF 74LS04 SOCKET.

4 28-PIN SOCKET PREPARATION:

BEND 1/16" OF END OF PINS 1, 14, 15, 27 OUT 90°

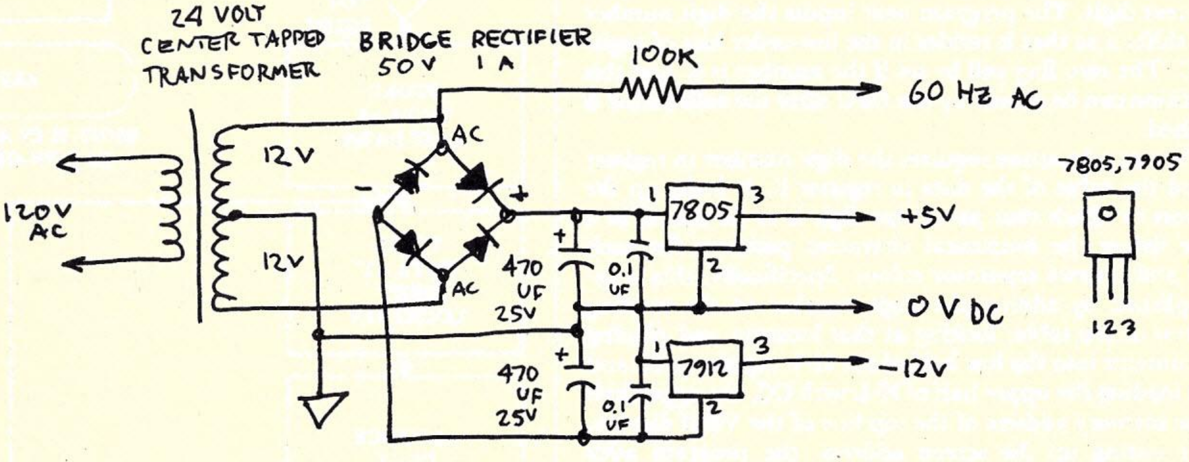
CUT PINS 6 – 13 TO 1/8"

CUT PIN 21, 28 TO 1/8"

BEND PINS 2 – 5, 16 – 20, 22 – 26 OUT 90° 1/16" FROM SOCKET BODY

MAKE TWO BENDS HALFWAY UP PIN 14 TO BRING ITS END OVER PIN 13

POWER SUPPLY


24 VOLT CENTER TAPPED TRANSFORMER
BRIDGE RECTIFIER 50 V 1 A
100K
60 Hz AC
120V AC
12V
12V
AC
AC
-
+
1
7805
3
+5V
470 UF 25V
0.1 UF
2
0 V DC
1
7912
3
-12V
470 UF 25V
0.1 UF
2
7805, 7905
1 2 3

Parts List for the Clock Power Supply

- 1 24-volt center-tapped transformer, 100 ma minimum
- 1 diode bridge rectifier, 50 PIV, 1 amp
- 2 470-microfarad 25-volt aluminum electrolytic capacitors
- 2 0.1-microfarad 25-volt ceramic or mylar capacitors
- 1 100K-ohm ¼-watt resistor
- 1 7805 integrated voltage regulator
- 1 7912 integrated voltage regulator
- Fuse, line cord, miscellaneous hardware

. . . And How It Runs

The test program displays the time on the upper-right-hand corner of the video display produced by a Processor Technology VDM-1 or Sol. The clock is connected to port FD (the parallel port on the Sol), and the listing shown is assembled to run in spare memory space in the Sol. The program will run on any 8080 machine with video display if it is changed to take account of the different port-assignment and display requirements. An assembly-language list is shown so that it may be reassembled as desired.

The program is self-contained and does not require a monitor program to operate. It uses register D to store the latest value of the seconds digit. If the program is to be used as a section of a larger program, this will have to be changed to use a memory location so that the data will not be lost in the shuffle.

In operation, the program enters at CAB4 and performs an initialization sequence. Colons are stored on the screen in the right places, and the digit is advanced until the zero digit (least significant seconds) is detected. At this point the program enters the data for that digit and compares it against the D register. If the two do not compare, it updates register D with the current value of the data and updates all the digits of the display. When it reaches the zero digit again, the program goes back to waiting for the next second.

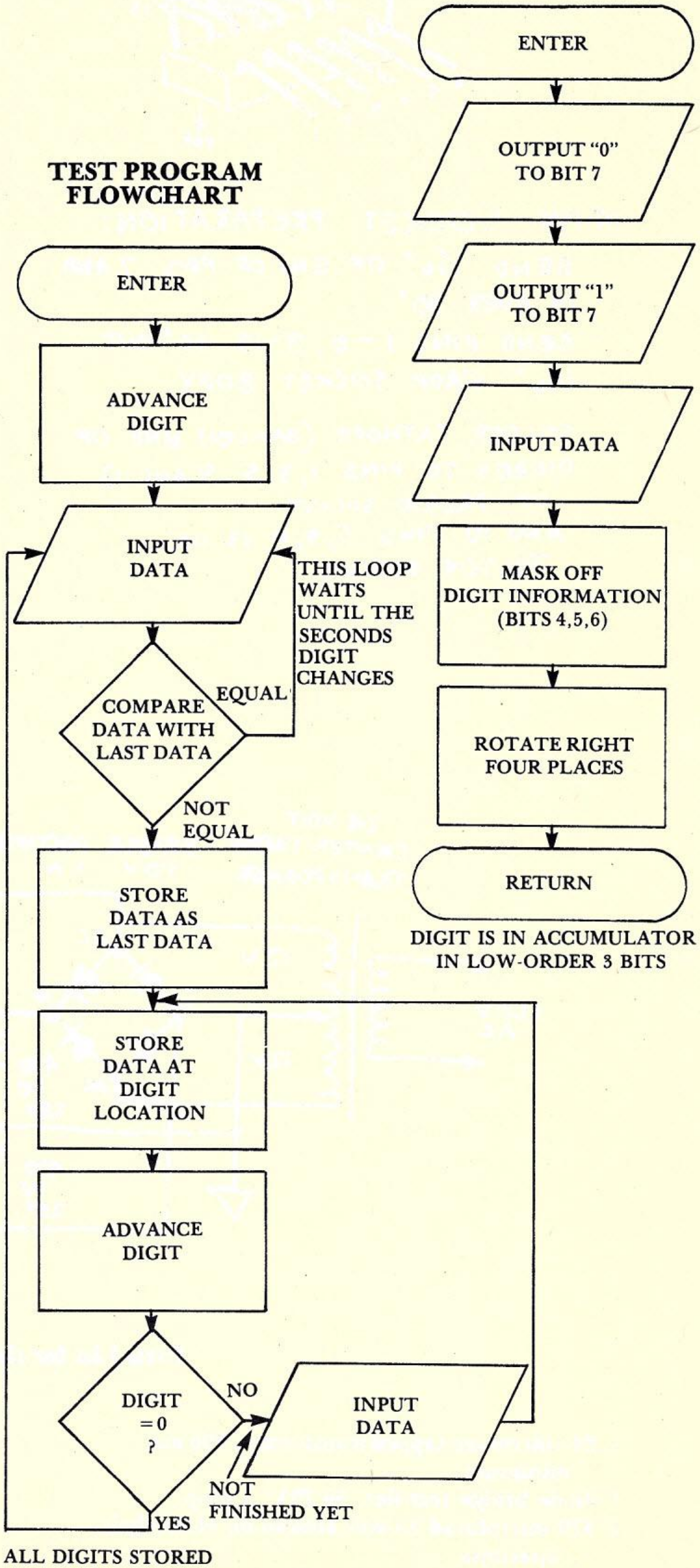
There are two subroutines: digit and out. The digit subroutine advances the digit count of the clock chip by sequentially outputting a *0* and a *1* on all bits of the output port. This causes the clock chip to see one cycle on the MPX TIMING input, and its internal counter will step to the next digit. The program next inputs the digit number and shifts it so that it resides in the low-order bits of register C. The zero flag will be set if the number is *0,* and this condition can be sensed by the CPU after the subroutine is finished.

The out subroutine requires the digit number in register C and the value of the data in register E. It looks up the address to which that particular digit is to be stored in a table listing the horizontal character positions for each digit and the two separator colons. Specifically, this is accomplished by adding the digit number to the starting address of the table, looking at that location and placing the contents into the low half of the H-L register pair, and then loading the upper half of H-L with CC, the upper half of the memory address of the top line of the VDM display. After setting up the screen address, the program adds (through the logical OR function) the necessary bits to make an ASCII numeric character out of the BCD data sitting in register E. Finally, it stores the result in the H-L address. If all goes well, a typical digital-clock numerical display will appear at the upper-right corner of the screen, changing once a second.

It must be noted that this program takes no account of the scrolling which the VDM-1 and the Sol perform. That would require some modifications, and would make a good project for learning how the software for those displays operates.

ADVANCE DIGIT SUBROUTINE

TEST PROGRAM FLOWCHART


ENTER
ADVANCE DIGIT
INPUT DATA
THIS LOOP WAITS UNTIL THE SECONDS DIGIT CHANGES
COMPARE DATA WITH LAST DATA
EQUAL
NOT EQUAL
STORE DATA AS LAST DATA
STORE DATA AT DIGIT LOCATION
ADVANCE DIGIT
DIGIT =0 ?
NO
INPUT DATA
NOT FINISHED YET
YES
ALL DIGITS STORED
ENTER
OUTPUT "0" TO BIT 7
OUTPUT "1" TO BIT 7
INPUT DATA
MASK OFF DIGIT INFORMATION (BITS 4,5,6)
ROTATE RIGHT FOUR PLACES
RETURN

DIGIT IS IN ACCUMULATOR IN LOW-ORDER 3 BITS

12:02
12:03
12:04
12:05
12:06
12:07
12:08
12:09
12:10
12:11
12:12
12:13
12:14
12:15
12:16
12:17
12:18
12:19
12:20
12:21
12:22
12:23

The Time-Testing Program

```
ADDRESS  CONTENTS  LABEL   MNEMONIC        COMMENTS

CAB4     1E 3A             MVI E,3AH       Move ASCII ":" to reg. E
CAB6     0E 07             MVI C,7         Move 7 to reg. C (digit no.)
CAB8     CD F0 CA          CALL OUT        Call output routine
CABB     0D                DCR C           Decrement digit no. (reg. C)
CABC     CD F0 CA          CALL OUT
CABF     CD E0 CA  BUMP    CALL DIGIT      Call digit advance routine
CAC2     C2 BF CA          JNZ BUMP        Loop and exit when digit = 0
CAC5     DB FD     WAIT    IN FD           Input data from port FD
CAC7     E6 0F             ANI 0F          Mask out upper nybble (half byte)
CAC9     BA                CMP D           Compare results against reg. D
CACA     CA C5 CA          JZ WAIT         If same, go to WAIT
CACD     5F                MOV E,A         Put data in reg. E
CACE     57                MOV D,A         Put data in reg. D
CACF     CD F0 CA  PUT     CALL OUT
CAD2     CD E0 CA          CALL DIGIT      Next digit
CAD5     CA C5 CA          JZ WAIT         If zero, go to WAIT
CAD8     DB FD             IN FD           Input data
CADA     E6 0F             ANI 0F          Mask out upper nybble
CADC     5F                MOV E,A         Put data in reg. E
CADD     C3 CF CA          JMP PUT         Jump to PUT
CAE0     AF        DIGIT   XRA A           Clear reg. A
CAE1     D3 FD             OUT FD          Output "0"
CAE3     2F                CMA             Complement (invert) accumulator
CAE4     D3 FD             OUT FD          Output "1"
CAE6     DB FD             IN FD           Input
CAE8     E6 70             ANI 70          Mask out lower nybble
CAEA     0F                RRC             Rotate accumulator right one bit
CAEB     0F                RRC
CAEC     0F                RRC
CAED     0F                RRC             Data is now in lower nybble
CAEE     4F                MOV C,A         Put data (digit no.) in reg. C
CAEF     C9                RET             Return from subroutine
CAF0     21 FE CA  OUT     LXI H, TABLE    Load H,L with table starting address
CAF3     AF                XRA A           Clear reg. A
CAF4     47                MOV B,A         Clear reg. B
CAF5     09                DAD B           Add B,C to H,L (C is table offset)
CAF6     6E                MOV L,M         Put table content in L
CAF7     26 CC             MVI H,0CCH      Put CC into H (VDM starting address)
CAF9     3E 30             MVI A, 30H      Put ASCII bias (30H) into A
CAFB     B3                ORA E           OR reg. A with reg. E
CAFC     77                MOV M,A         Store A at (H,L) in memory
CAFD     C9                RET             Return from subroutine
CAFE     3F        TABLE   DB 3F           Data; 3F (rightmost VDM char. address)
CAFF     3E                DB 3E
CB00     3C                DB 3C
CB01     3B                DB 3B
CB02     39                DB 39
CB03     38                DB 38
CB04     3D                DB 3D           Right-hand colon char. address
CB05     3A                DB 3A           Left-hand colon char. address
```

Things

Six years ago, in my book *What Computers Can't Do,* I accused AI workers of covering up their failures with ad hoc gimmicks and rhetorical claims. Nothing has happened since that time to change my view that, as far as understanding intelligence is concerned, the field is stagnating. But there has been a surprising improvement in chess-playing programs, as well as an encouraging new tendency of some to face up to the underlying unsolved problems involved in understanding natural language.

My earlier outrage at the misleading names given to programs such as Newell, Shaw, and Simon's GPS (General Problems Solver) is now shared by some workers in the field. Drew McDermott of the M.I.T. Artificial Intelligence Laboratory defended and extended my critique in the April 1976 SIGART *Newsletter:*

> [I]n AI, our programs to a great degree are problems rather than solutions. If a researcher tries to write an "understanding" program, it isn't because he has thought of a better way of implementing this well-understood task, but because he hopes he can come closer to writing the *first* implementation. If he calls the main loop of his program "UNDERSTANDING," he is (until proven innocent) merely begging the question. He may mislead a lot of people, most prominently himself, and enrage a lot of others.

McDermott also singled out poor old GPS:

> Many instructive examples of wishful mnemonics by AI researchers come to mind once you see the point. Remember GPS? By now, "GPS" is a colorless term denoting a particularly stupid program to solve puzzles. But it originally meant "General Problem Solver," which caused everybody a lot of needless excitment and distraction. It should have been called LFGNS—"Local Feature-Guided Network Searcher."

My earliest accusation, made over a decade ago, that work in AI resembles alchemy more than science has also been accepted—albeit with a new upbeat twist:

> In some ways, [AI] is akin to medieval alchemy. We are at the stage of pouring together different combinations of substances and seeing what happens, not yet having developed satisfactory theories. This analogy was proposed by Dreyfus (1965) as a condemnation of artificial intelligence, but its aptness need not imply his negative evaluation...it was the practical experience and curiosity of alchemists which provided the wealth of data from which a scientific theory of chemistry could be developed.

Terry Winograd (from whose memorandom to the Stanford Artificial Intelligence Laboratory in May of last year this quote is extracted) has a point. As long as researchers in AI admit and learn from their failures, their work may in the end provide techniques and data for a more promising approach. But admitting their failures so that others can learn from their mistakes—an essential part of any scientific field—is still virtually unknown in AI circles. McDermott agrees that "AI as a field is starving for a few carefully documented failures." And he warns, "Remember, though, if we can't criticize ourselves, someone else will save us the trouble." I take this as my cue to return for another critical look at the field.

AI as a field is starving for a few carefully documented failures.

Two issues agitate the private computer communication networks and overflow into the public press: there is jubilation over the success of CHESS 4.5, and there is acrimonious debate over the correct strategy for language-understanding programs. Chess programs have made remarkable progress during the past two years, to the point where they can hold their own against experts. Language understanding, on the other hand, despite a new flurry of activity, is still in the same state of stagnation it was in as of 1972, and although this has led some researchers to sober thoughts on the difficulty of programming human understanding, it has led others to ever more extravagant promises and claims. In order to form a reasonable opinion about what has and can be done, we must begin by looking in more detail at the computer's successes at chess and the stagflation afflicting language understanding.

In 1957 Simon predicted that a computer would be world chess champion in ten years. Writing eight years later, and reviewing the chess programs written up to then, I concluded: "Still no chess program can play even amateur chess, and the world championship is only two years away." That line was later quoted out of context by Toffler, Simon, and others to show that I thought no chess program could *ever* play even amateur chess. This was never my view. I see no in-principle barrier to a computer becoming world champion, but I do think that compu-

Computers *Still* Can't Do

by Hubert L. Dreyfus

ters play chess quite inefficiently compared to human beings, and I must admit that I was greatly surprised when, in July 1976, the Northwestern University program CHESS 4.5 won the class B section of the Paul Masson American Chess Championship with an impressive five wins, no losses, and then went on in February 1977 to win the 84th Minnesota Open Tournament against experts and high-class A players.

The excitement in the computer chess world centers on whether this new breakthrough in chess playing will save the honor (and money) of those who ten years ago took International Master David Levy's bet that he would not lose to a computer program before 31 August 1978. Levy is not worried; he points out that the "expert" rating which CHESS 4.5 has earned is an overestimate owing to the inexperience of human opponents in coping with the special strengths and weaknesses of machine play. I'm still ready to bet on Levy's side. In judging the power of the program, it is important to know that its strength is based on the brute force examination of around five hundred thousand look-ahead positions; a human master can often play better chess looking at fewer than fifty moves.

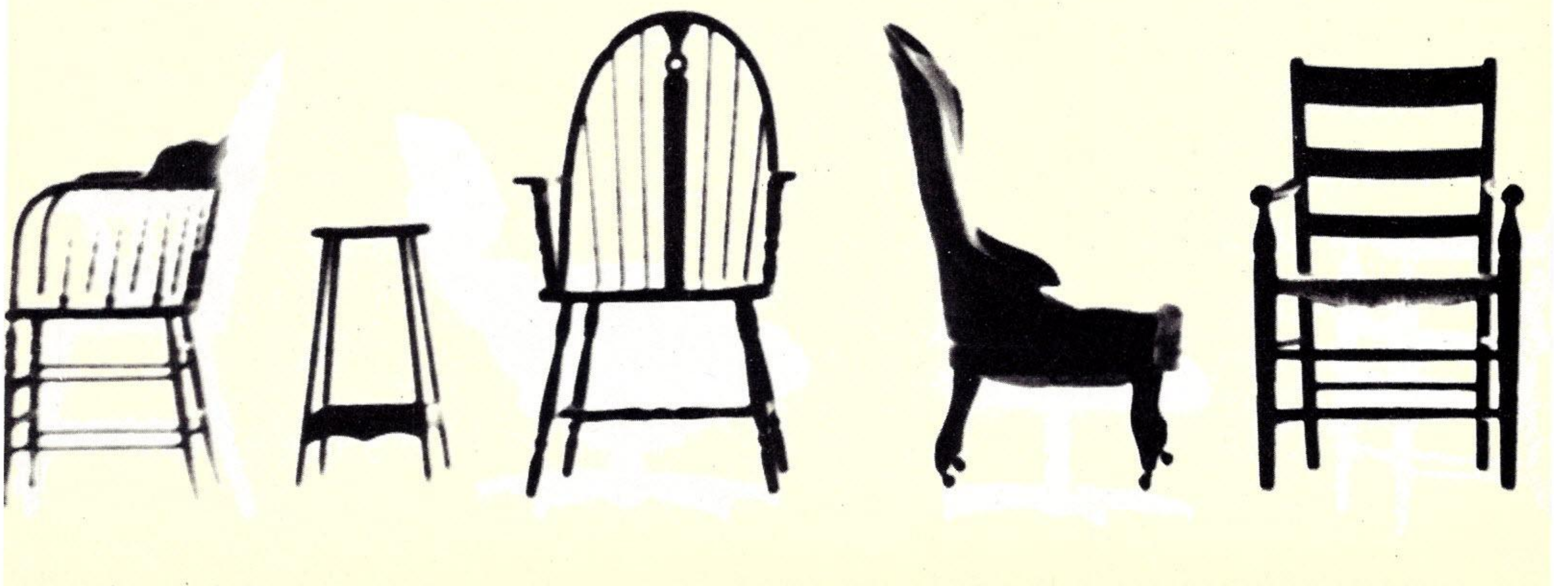
Human players obviously do not count out large numbers of alternatives. They have a sense of the game as a developing situation, and by playing over thousands of book games they presumably develop chess intuition, which enables them to recognize a present position as more or less similar to a position in some previous game that turned out to be dangerous or promising. At present, computers using exhaustive search and masters using selective search can both look ahead about six or seven plays. Given the exponential growth of alternative moves, it will not be possible, without better tree-searching heuristics, to significantly increase the computer's power to look ahead. Thus, with present programs, what is really at stake is how far computer programs—which must use context-free attributes such as "material balance" (where a numerical value is assigned to each piece on the board and the total score is computed for each player) or "center control" (where the pieces bearing on each centrally located square are counted) in making their evaluations—can make up by sheer brute force for the economy of move consideration and long-range strategy which goes with seeing similarity to other games.

It is important to see that some sort of template matching won't help to account for a master player's ability to recognize a present board configuration as more or less like a situation that has occurred in an earlier game. It is astronomically unlikely that two positions will ever turn out to be *identical,* so what has to be compared is *similar* positions. But similarity is obviously not just having a large number of pieces on identical squares. Seeing the aspects which make two positions *similar* is exactly what requires a deep understanding of the game.

Could the computer, with more selective search heuristics, calculate far enough ahead to make up for the fact that it must follow rules and evaluate positions on the basis of simple attributes like the poorest potzer? That remains to be seen. The Levy match, now just a few months away, will provide an interesting test. But for those interested in the future of AI, it is important to bear in mind that whatever the outcome, chess is just a game and so, in principle at least, can be played perfectly by machine. The real areas of difficulty where I predicted failure lie elsewhere.

Games have always been a favorite field for AI work because games are by definition cut off from the real world. In a game, what is relevant is delimited beforehand. Thus, in chess, we know that the color and position of a piece are always relevant; its temperature, how much it weighs, what it is made of, and so on can never be relevant. All games are self-contained toy worlds carefully isolated from everyday human activities.

Computer researchers have carried over this reassuring aspect of games in



Photographs by Thomas Hall

creating what they call microworlds, in which, as in a game, what can possibly be relevant is determined beforehand. Thus we have Winograd's language-understanding program, SHRDLU, which is restricted to a microworld of colored blocks which can be moved in certain restricted ways, or Winston's so-called concept-learning program, which "learns" concepts such as *arch* by combining a restricted set of primitives such as *left-of, above, supported-by,* and *standing* provided in advance by the programmer. Since what is essential to natural-language understanding and to learning is that the performer *discover* what facts about our world are relevant in a particular situation, such microworld programs cannot be considered contributions to the simulation and understanding of everyday human intelligence.

Moreover, since each such program depends on predigested attributes which work only in one specific situation, it cannot be generalized. This ad hoc character of all microworld programs is beginning to be recognized by some AI researchers. As usual, the acknowledgement can be found, not in the direct admission of failures, but in the prologue to some new program proclaimed to be the answer to the belatedly admitted limitations of the previous year's breakthrough. Thus Winograd, in his 1972 presentation in *Cognitive Psychology* of his blocks-world program, included a criticism of the ad hoc character of previous programs (which I also had criticized) such as Bobrow's STUDENT: "Their restricted domain often allows them to use special purpose heuristics which achieve impressive results with a minimum of concern for the complexities of language." SHRDLU, in turn, was obliquely criticized by Winograd and Bobrow in the opening presentation of their new knowledge-representation language (KRL), which appeared in the January 1977 issue of *Cognitive Science:*

> There is currently no suitable base on which to build sophisticated systems and theories of language understanding. . . . Current systems, even the best ones, often resemble a house of cards. The researchers are interested in the higher levels, and try to build up the minimum of supporting props at the lower levels. . . . The result is an extremely fragile structure, which may reach impressive heights, but collapses immediately if swayed in the slightest from the specific domain (often even the specific examples) for which it was built.

KRL, of course, seems to have put these ad hoc limitations behind it. But before we consider KRL, we must look at Minsky's theory of frames on which it is based.

In his classic paper "A Framework for Representing Knowledge," published in 1975 in *The Psychology of Computer Vision,* Minsky proposed an abstract data structure for representing everyday knowledge in terms of "stereotyped situations." This formal representation is similar to what the phenomenologist Edmund Husserl once called *noema.* In *What Computers Can't Do,* I contrasted Husserl's conception with the passive model of intelligence as the receiving and combining of facts, the view then popular in AI. For Husserl the noema for any type of object has, among other structural elements, a nucleus containing symbolic representations of all the features which can be expected with certainty in exploring a certain type of object, plus *predelineations* of those features which are possible but not necessary aspects of that type of object. Intelligence is thus understood as the ability to *search* for anticipated facts, rather than as merely the analyzing of whatever data is received.

In Minsky's model of a frame, the "top level" is a developed version of what Husserl called the noematic nucleus, and Husserl's predelineations have been made precise as "default assignments"—additional features that can normally be expected. The result is a step forward in AI from a passive model of information processing to one which tries to take account of the complex interactions between a knower and his world.

Husserl thought of his method as the beginning of progress towards a rigorous science of philosophy, and Patrick Winston, head of the M.I.T. Artificial Intelligence Laboratory, hailed Minsky's proposal as "the ancestor of a wave of progress in AI." But Husserl's project ran into serious trouble and there are signs that Minsky's may, too.

After twenty years of trying to spell out the components of the noema of a simple object, Husserl found that his analysis had to include more and more of a subject's total knowledge of the world, and he sadly concluded, at the age of seventy-five, that he was a "perpetual beginner" and that phenomenology was "an infinite task." Similarly, there are hints in an unpublished draft of the frame paper that Minsky is headed for the same problems that finally overwhelmed Husserl.

> Just constructing a knowledge base is a major intellectual research problem. . . . We still know far too little about the contents and structure of common-sense knowledge. A "minimal" common-sense system must "know" something about cause-effect, time, purpose, locality, process, and types of knowledge. . . . We need a serious epistemological research effort in this area.

To quote Minsky, "One should choose carefully the chair-description frames that are to be the major capitols of chairland."

Minsky's naivete and faith are truly touching. Philosophers from Plato to Husserl, who uncovered all these problems and more, carried on serious epistemological research in this area for two thousand years without notable success. Moreover, the list Minsky includes in this passage deals only with natural objects and their positions and interactions. As I argued in my book, intelligent behavior also presupposes a background of social practices and instituitions. There are observations in the frames paper that show that Minsky understood my point:

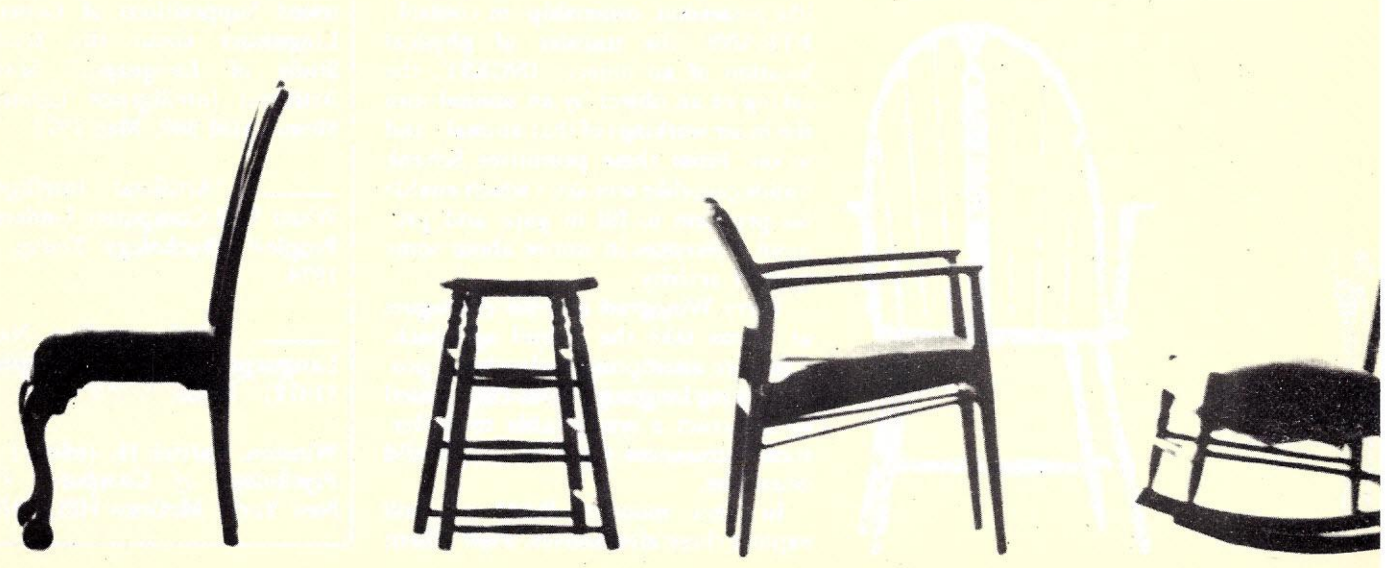
> Trading normally occurs in a social context of law, trust, and convention. Unless we also represent these other facts, most trade transactions will be almost meaningless.

But Minsky seems oblivious to the hand-waving optimism of his proposal that programmers rush in where contemporary philosophers such as Wittgenstein and Heidegger have feared to tread and simply make explicit the totality of human practices which pervade our lives as water encompasses the life of a fish.

To make this essential point clear, it helps to take an example used by Minsky and to look at what is involved in understanding a piece of everyday equipment as simple as a chair. No piece of equipment makes sense by itself. The physical object which is a chair can be defined in isolation as a collection of atoms, or of wood or metal components, but what makes it a *chair,* what makes possible its role as equipment for sitting, is its place in a total practical context. This presupposes certain facts about human beings—fatigue, the ways the body bends—a network of other culturally determined equipment—tables, floors, lamps—and skills—eating, writing, going to conferences, giving lectures. Chairs would not be equipment for sitting in traditional Japan or the Australian bush.

Anyone in our culture understands such things as how to sit on kitchen chairs, swivel chairs, and folding chairs, and how to sit in armchairs, rocking chairs, deck chairs, barber's chairs, sedan chairs, dentist's chairs, basket chairs, reclining chairs, wheelchairs, sling chairs, and beanbag chairs—as well as how to get out of them again. This would seem to involve a repertoire of bodily skills which is indefinitely large, since there seems to be an indefinitely large variety of chairs and of successful—graceful, comfortable, secure, poised—ways to sit in them. Moreover, understanding chairs also includes social skills such as being able to sit appropriately—sedately, demurely, naturally, casually, sloppily, provocatively—at dinners, interviews, desk jobs, lectures, auditions, and concerts (intimate enough for there to be chairs rather than seats), and in waiting rooms, living rooms, bedrooms, courts, libraries, and bars (of the sort sporting chairs, not stools).

In the light of this amazing capacity, Minsky's remarks on chairs in his frame paper seem more like a review of



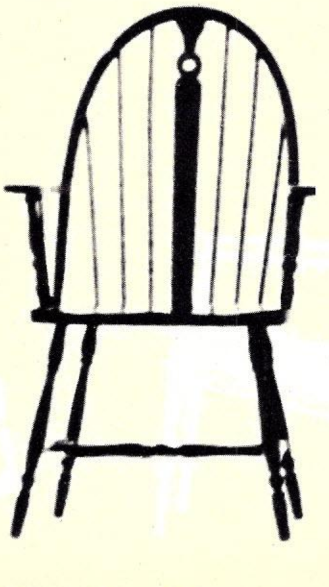
the difficulties than even a hint of how AI could begin to deal with our common-sense understanding in this area.

> There are many forms of chairs, for example, and one should choose carefully the chair-description frames that are to be the major capitols of chair-land. These are used for rapid matching and assigning priorities to the various differences. The lower priority features of the cluster center then serve . . . as properties of the chair types. . . .

There is no argument why we should expect to find elementary context-free features characterizing a chair type, nor is there any suggestion as to what these features might be. They certainly cannot be legs, back, seat, and the like, since these are aspects of already recognized chairs, not context-free features, defined apart from chairs, which then cluster in a chair representation. Minsky continues:

> Difference could be functional as well as geometric. Thus, after rejecting a first try at "chair" one might try the functional idea of "something one can sit on" to explain an unconventional form.

But a function so defined is not abstractable from human embodied know-how and cultural practices. If it is treated merely as an additional symbolic descriptor along with physical features, function cannot even distinguish conventional chairs from saddles, thrones, and toilets. Minsky concludes:



> Of course, that analysis would fail to capture toy chairs, or chairs of such ornamental delicacy that their actual use would be unthinkable. These would be better handled by the method of excuses, in which one would bypass the usual geometrical or functional explanation in favor of responding to contexts involving art or play.

This is what is required, all right, but by what elementary features are these contexts to be recognized? There is no reason at all to suppose that one can avoid the difficulty of formally representing our knowledge of chairs by abstractly representing even more holistic, concrete, culturally determined, and loosely organized human practices such as art and play.

It is to Minsky's credit that he has at last brought these problems, familiar to phenomenologists, into the open in AI, and that he has provided a model so vague and suggestive that it can be developed in several different directions. As one would expect, two alternatives immediately present themselves: either to use frames to deal with common-sense knowledge as if everyday activity were a microworld, or to try to develop frame structures capable of capturing the open-ended character of everyday life.

Of the two most influential current schools in AI, one, that of Roger Schank and his students at Yale, has tried the first approach. Schank and his followers propose a set of twelve basic actions such as ATRANS, the transfer of an abstract relationship like possession, ownership, or control; PTRANS, the transfer of physical location of an object; INGEST, the taking of an object by an animal into the inner workings of that animal; and so on. From these primitives Schank builds gamelike scenarios which enable his program to fill in gaps and pronoun references in stories about some specific activity.

Terry Winograd and his colleagues at Xerox take the second approach. They are attempting to develop a programming language which can be used to construct a system able to understand utterances made in real world situations.

In next month's *ROM* we will explore these alternatives, their salient features and problems, and some of their repercussions. We may then be better prepared to say whether AI is still stuck or whether, between Schank's restaurant game and Winograd's top-down holism, progress is being made in bringing AI and man to a mutual understanding. ▼

Further Reading

Bobrow, D.G., and Winograd, T. "An Overview of KRL, a Knowledge Representation Language." *Cognitive Science,* January 1977.

Dresher, B. Elan, and Hornstein, N. "On Some Supposed Contributions of Artificial Intelligence to the Scientific Study of Language." *Cognition* 4 (1976): 321-398.

Dreyfus, Hubert L. *What Computers Can't Do.* New York: Harper & Row, 1972.

Society for the Interdisciplinary Study of The Mind. Symposium for Philosophy and Computer Technology. State University College, New Paltz, New York (March 1977).

Schank, Roger C. "The Primitive Acts of Conceptual Dependency." *Theoretical Issues in Natural Language Processing* (June 1975).

Warren, N. (editor). *Advances in Cross-Cultural Psychology* (Vol. 1). London: Academic Press, 1978.

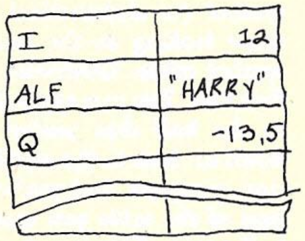
Winograd, Terry. "On Some Contested Suppositions of Generative Linguistics about the Scientific Study of Language." Stanford Artificial Intelligence Laboratory Memo AIM-300, May 1977.

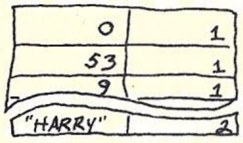
———. "Artificial Intelligence: When Will Computers Understand People?" *Psychology Today,* May 1974.

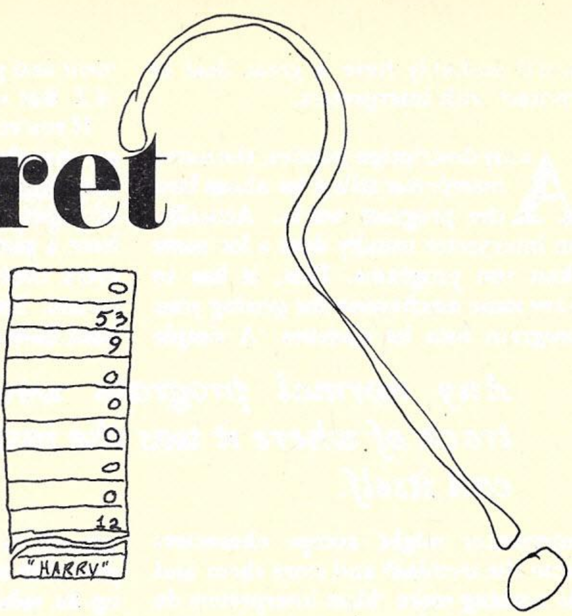
———. "Understanding Natural Language." *Cognitive Psychology* 3 (1972): 1-191.

Winston, Patrick H. (editor). *The Psychology of Computer Vison.* New York: McGraw Hill, 1975.

Why Interpret







by Eben Ostby

There's grape juice in the central processor, a wishbone in the teletype, and you still want to try out the program?

Suppose you want to try out a new idea for a program you just thought up. You'd probably write down the basic idea in a general fashion, perhaps in a slightly mixed-up version of BASIC. Then you'd program the thing in BASIC and run it on your computer, right?

Okay, your nephew just spilled grape juice into the central processor, and there's a wishbone lodged in the teletype. You get the idea: the computer's disabled. You *still* want to try out the program. Now what do you do? Well, I'd grab some convenient friend (preferably not your nephew) and play computer for a while. Playing computer isn't hard—it just means executing each instruction in your program yourself—but it is pretty boring.

One of you can read off the BASIC instructions, and the other can "execute" them. And if you want your computer-playing to be really accurate, you should try the different ways of executing BASIC programs that computers use. Some computers use BASIC *compilers.* A compiler is a program that takes, as input, your BASIC program and turns it into assembly-language instructions that can be executed directly on your computer.

Here's how you and your friend would compile a program. First, you must come up with a set of primitive instructions. These should include the simplest elements that go into any program—you'll want to be able to add, subtract, multiply, divide, input and output numbers and letters, test if numbers are equal to or less than other numbers, and "jump" to other parts of the program. Next, you've got to read off each BASIC statement and turn it into these primitive instructions. And finally, you'll want to execute the primitive instructions that you just created. You read them off, in turn, while your friend does what you specify. The only time your friend needs to tell you what to do is when you encounter a "jump" statement. Then he tells you which statement you are to read next.

Does this sound like an awkward way to run a program? Well, it is. But it's also the most efficient (and most popular) way to run many programs. When a program is going to be run many times, the first part—translating the program into the primitive machine instructions—only has to be done once. The last part is inherently fast on a computer (although you probably found it slow for people). However, the translating process is difficult.

May I suggest a more straightforward way? Why not read the individual BASIC statements to your friend and have him execute the entire statement, right then. This is probably the first method you thought of, anyway, and is called *interpreting* the program. Many home computers have BASIC interpreters rather than BASIC compilers. Even though they sound simpler, interpreters are slower than compilers most of the time, because they have to figure out what each BASIC statement does, each time they encounter it. If you write a loop that executes a single statement 100 times, the interpreter has to decode that statement each of the 100 times through the loop. In the end, more time is spent figuring out what each statement means than is spent actually executing the statements.

So you've got a trade-off: compilers are more efficient, but interpreters are simpler. But since you will presumably be running very few "monster" programs that eat up computer time,

Diagrams by the author,
query by Korkie

you'll probably have a great deal of contact with interpreters.

As my description implies, the name *interpreter* tells a lot about how the program works. Actually, an interpreter usually does a lot more than run programs. First, it has to have some mechanism for getting your program into its clutches. A simple interpreter might accept characters from the terminal and store them and do nothing more. Most interpreters do much more than that, though. The things you're most likely going to want are facilities to change parts of program lines, print out lines, and perhaps search for certain things in your program. Already you've got the makings of a text editor, and you haven't even started interpreting yet!

Any normal program would have lost track of where it was the instant it tried to call itself.

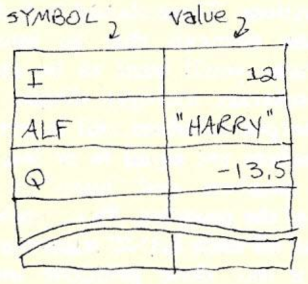
Somehow, then, you've got to get your program into the interpreter. Once it's in there, what happens? The process is pretty much like what you and your friend did a few paragraphs ago. Take, for example, a BASIC program that loops:

```
0100 LET I=0
0110 PRINT I;
0120 LET I=I+1
0130 IF I<20 THEN 110
0140 STOP
```

An interpreter would store this program pretty much verbatim. (In some cases, the interpreter reads the program as it is entered and stores line numbers as binary integers, just to speed things up later.) When you enter *GO* at the keyboard, BASIC finds the first statement. It looks at the first word, *LET,* and realizes that the rest of the statement is going to compute something and put it somewhere. So the next thing it needs to do is hand over the rest of the statement to a special routine that decodes arithmetic statements. That routine figures out that *I* is to be set to zero, does just that, and the main part of the interpreter then takes over again. Looking at statement 110, it sees the keyword *PRINT,* so it can call upon the special routine that does nothing but printing. That routine looks at the rest of the statement and properly prints out the value of *I.* But what's *I?*

If you're playing computer, you can say that the variable *I* is represented by the upper-left-hand corner of a piece of paper. But your computer doesn't have a piece of paper it can glance at every time a reference to a variable occurs. Somewhere in its memory it must have a list of all the variables that you use in your program. Every time you refer to a variable, it has to look up its value in this table. This table, called a symbol table, is crucial to the operation of the interpreter. For instance, in the tiny program above, there must be no fewer than five references to the symbol table. Normal-sized programs might require hundreds of symbol-table references. That's why many interpreters have elaborate (and very fast) routines for finding things in the symbol table.

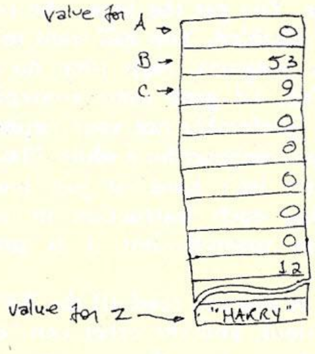
The simplest symbol table just has the symbols stored in the order they were encountered by the interpreter. It might look something like this:



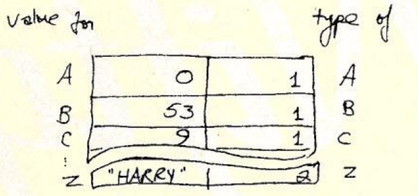
This is a very easy way to store things, because you can just insert each new item after the last one in the list, anytime you have to. But when you want to find something in the list, it's not so easy. Each time you want to find the value of a variable, you have to start at the beginning and check each entry in turn, until you find the one you want. Since you only store a symbol's name in the table once, and you probably want to find its value or change its value many times, this can take a long time. In fact, if there are 100 symbols in your symbol table (not uncommon in large programs), the average number of times you'll have to check an entry to find just one symbol is fifty times! Computers are fast, but not so fast that you'll want to wait around for *that.*

Another way you might set up the symbol table is by ordering the names, or putting them in alphabetical order. Then you can make use of that fact when you want to find a symbol. You can do it like this. Check the middle symbol in the table. If the symbol you're looking for is *after* the middle symbol (in alphabetical order), continue looking in the last half of the symbol table. Otherwise, look in the first half. You can apply the same trick to the half that you've narrowed it down to: look in the middle of the section, compare, and you've reduced the part of the table you must look in by half again. This is a very fast way of looking. If you have 100 entries in the table, on average, it will take only seven comparisons to find any particular symbol! But with this method, putting something into the table is a pain—you have to sort the table every time you put something into it.

Suppose you are limited to twenty-six different variables, with the names A through Z. In this case, you can turn the letter (the name of the variable) into a number and use it as a "pointer" into the symbol table, meaning you don't have to do any comparisons at all. For instance, if you've got the letter *I* in your program, you know to look in the ninth spot in the symbol table for the value of *I.* This symbol table looks like:



Actually, there's more to it than this, though. Since most programs refer to numbers and character data, there has to be some part of the symbol table that indicates which kind of data the variable happens to hold. In this new, improved table, there is another number included with the value of each symbol: the type number.



Since this example uses just two types—number and character—let a 1 indicate a number, and a 2 indicate a character variable. (Most languages have at least three types of data: character, integer, and real number. Many also have Boolean numbers—numbers that can only be 0 or 1, and pointers—which hold the addresses of other variables. Some languages, like ALGOL and PL/I, have scores of data types; in some, you can define your own.)

Anyway, there still is a problem with this symbol table: you're limited to twenty-six different variables with the names A through Z. One way you might try to fix this up would be taking the name of the variable—any name this time—treating it as a number, and dividing it by the size of the symbol table. The remainder, after you do the division, will be a number that is between 0 and the size of the table. Then you can treat this number as you did the single letter in the previous table: the number becomes a pointer into the symbol table. Sometimes you'll find that, after doing the division, two different variable names will yield the same number. When this happens, you can simply allot the correct space in the symbol table to the first variable that comes along and put the second one in the first free space in the table. This table is pretty fast—usually you can find the variable you want without doing any searching at all; at worst, you'll just have to do a little bit of searching. This technique is known as *hashing*.

The expression analyzer must be able to ask itself questions—and get answers!

The symbol table is not the only important part of an interpreter. The biggest part of the interpreter is known as an *expression analyzer*. For most languages, the expression analyzer is the part of the interpreter that figures out what's to the right of the equal sign in an assignment statement. But, since arithmetic expressions can occur in many different places in a program—*PRINT* statements, *INPUT* statements, *DO* loops, and *IF* statements—the expression analyzer will be called into use wherever there's an arithmetic expression. It's actually a pretty complicated beast. It has to be able to recognize the different parts of an arithmetic expression: numbers, signs, operations (*, +, −, /), variable names, names of functions, and special symbols (parentheses, commas, and so on). But most important of all, it has to know what to do with them. You can probably see that it must call on the symbol-table routine to find out the values of variables, but it really must have the ability to ask itself questions—and get answers!

It works like this: suppose you have the expression

5*(4−7)

the expression analyzer first sees that it has to multiply two numbers together—something it can do quite well. It picks out the 5, as the first number, and then goes on to the next number. But the next number is in parentheses, so it doesn't have a ready answer. Since the number in parentheses is itself an expression, the expression analyzer calls the expression analyzer (itself, right?) and asks for the value of the expression

4−7

This it can do easily, so it does, and returns the result −3. The program that called the analyzer here was the analyzer, so it uses the result, multiplying it by the 5, to get −15.

The special ability required here is not to get lost. Any normal program would have lost track of where it was the instant it tried to call itself. But the analyzer knows enough to save all current information—where in the computer it is executing, what expression it is working on, and the values of important variables—before it calls itself. Then, when it returns, it can just restore those values to their proper places and continue. This is known as *recursion*.

Really, then, all that is needed now to create a working interpreter is something to tie all these parts together: a program that will do things like start up the user program when *GO* is typed on the keyboard, call the expression analyzer when the program has a *LET* statement in it, check the values of the variables in an *IF* statement, and keep track of which statement to execute when. Assuming, of course, you know what language you want to interpret.

There are some languages that are particularly good for interpreter implementation. BASIC is a good example, because an interpreter lets you see more of what's going on in your programs. If the interpreter is well planned, you can stop the program, examine the contents of the variables, and restart again; however, with compiled programs, these functions are usually more difficult. APL is also good for interpreters, because computers spend more time executing APL programs than they do figuring out what the programs mean. Since the inefficient part of interpreters is in the decoding, or figuring out of the programs, APL is less inefficient for interpreting than most languages. Yet another good language for interpreting is LISP (see next page).

Once you've put it all together, you'll have an interpreter, that handy little program that operates directly on the source program in memory. It's a lot simpler than interpreting Polish for President Carter, and it will get you out of trouble, not into it. ▼

CRYPTOGRAM SOLUTIONS

1. DEVELOPMENT OF CRYPTANALYTIC MACHINES IN THE UNITED STATES DATES FROM THE EARLY THIRTIES AND THE USE OF I.B.M. TABULATORS BY THE U.S. NAVY.

2. COULD EARLY HOLLERITH CENSUS MACHINES HAVE PERFORMED FREQUENCY COUNTS AND OTHER STATISTICAL OPERATIONS? YES, INDEED!

Lisp Is a Nifty

LISP is a nifty language that has just begun to be found on micros. Unlike most other languages you have probably seen, LISP couldn't care less about numbers. In fact, LISP can do only a very few things to one strange sort of data structure, called a List; it can do nothing else. But it happens that LISP is one of the most powerful, interesting computer languages in existence.

As I said, LISP data are a form of list structure. The basic list is called an S-expression. S-expressions are made up of *atoms,* each of which must start with a letter and can contain many letters and digits. In other words, an atom is formed the way a variable name is formed in most other languages. Here are some examples of atoms:

A WEASEL R2D2 ATOMICSYMBOL

LISP treats atoms as individual units—they can't be broken up into their component letters.

S-expressions are put together in a very specific way. In particular, an S-expression can be either an atom, or two smaller S-expressions with a dot between them. The following are some S-expressions that were made from the atoms shown above:

A
(A. WEASEL)
((R2D2.A).ATOMICSYMBOL)
((R2D2.A).(A.(WEASEL.ATOMICSYMBOL)))

LISP has a couple of special atoms, too. They are

F, T, and *NIL*

F and *T* are used to represent *false* and *true,* respectively. *NIL* has a special purpose. In LISP, many S-expressions represent what are known as Lists (with a capital *L*). A List is just a string of atoms, written with parentheses around it, and looks like

(A R2D2 WEASEL A ATOMICSYMBOL)

But a List is represented in LISP by an S-expression that looks like this:

(A.(R2D2.(WEASEL.(A.(ATOMICSYMBOL. NIL)))))

The *NIL* is stuck in at the end to fill out the S-expression. A List can also have Lists within it, as in the S-expression

(A R2D2 (A WEASEL) ATOMICSYMBOL)

which is the same as

(A.(R2D2.((A.(WEASEL.NIL)). (ATOMICSYMBOL.NIL))))

Now you know what S-expressions and Lists are. LISP can only do a few select things to them. One thing it can do is stick two Lists (either kind) together to make a new one. In LISP, programs are written as if they themselves were Lists—in this case, to stick two Lists called *A* and *B* together, you would write

(CONS A B)

which says "consolidate *A* and *B*." If *A* is the List

(A R2D2)

and *B* is the atom *WEASEL,* then the result of *(CONS A B)* is the S-expression

((A.(R2D2.NIL)).WEASEL)

If *A* is jut the atom *A,* and *B* is just the atom *B,* then the result of *(CONS A B)* is *(A.B)*. All *CONS* does is stick two things together with a dot between them. If the result can be written as a List (with no dots), you may by all means do so. For instance, the result of *(CONS A B)*, where *A* is the atom *A* and *B* is the S-expression *(B.NIL)*, is just the List *(A B)*.

LISP can also take apart S-expressions and Lists. If you have an S-expression, you can get the first element of it with the function

(CAR A)

If *A* is the List *(A B C)*, then *(CAR A)* is the first part of the List, or the atom *A*. If *A* is the S-expression *((A.B).C)*, then *(CAR A)* is *(A.B)*. There's also a function that gives you the last part of the List; it's called *CDR*. (The reason why the functions have such odd names has to do with the computer on which LISP was first run.) If *A* is the List *(A B C)* again, then *(CDR A)* gives you the List *(B C)*—the last part of the

Language...

by Eben Ostby

argument. You can see why if you write the list as an S-expression. *A* is written (*A*. (*B*. (*C*.*NIL*))). The second half of the S-expression is what is to the right of the first dot, or (*B*. (*C*.*NIL*)). Written as a List, this is just (*B C*). If *A* was the S-expression ((*A*.*B*).*C*), then the *CDR* of *A* is just the atom *C*.

There are only two other operations that LISP can do. One is testing to see if two atoms are equal. The function that does this is called *EQ*. *EQ* returns a *T* (if the two atoms are equal) or an *F* (if they're not). If you give *EQ* arguments that are not atoms, LISP doesn't know what to do. (*EQ A A*) is true, as long as *A* is an atom. If *A* is the atom *WEASEL,* and *B* is also the atom *WEASEL,* then (*EQ A B*) has the value *T*. If *B* is the atom *R2D2,* then (*EQ A B*) has the value *F*. The other operation is testing to see if something is an atom. (*ATOM A*) returns *T* if *A* is an atom, and *F* if it is not.

Finally, LISP has a way of putting these things together to form a program. First, there is a function called *COND* which evaluates a kind of *IF* statement. It works like this. Its arguments are Lists with two parts. *COND* evaluates the first part of the List. If it has the value *T,* then the result of the *COND* statement is the result of the second part of the List; otherwise, *COND* tries the second List in its argument. For instance, the expression

```
(COND((EQ A B) A) (T B))
```

causes *COND* to evaluate (*EQ A B*). If *A* is equal to *B,* then *COND* returns the value of *A*. Otherwise, it goes on to the next part. Since the value of the first part of that List is *T, COND* will return the *B*. Note that anytime a function is called, it and its arguments have to be put in parentheses within the outer parentheses—that's why (*EQ A B*) is in parentheses, while *T* is not.

The only other thing needed to make a LISP program work is a way to define it to the interpreter. For this, there are two reserved words, *DEFINE* and *LAMBDA*. Entering (*DEFINE*(....)), where the dots are my function, will cause LISP to define the function. But the function itself must look like this:

```
(NAME(LAMBDA(A B C)...))
```

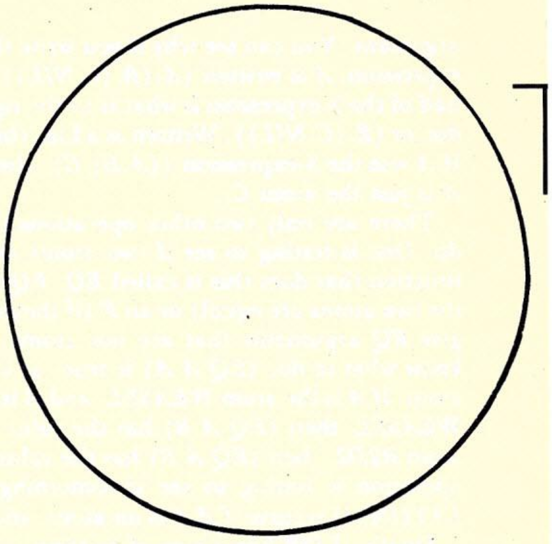
NAME is the name of the function, and *LAMBDA* alerts LISP to the fact that the next thing in parentheses is a List of the names of the parameters.

Why, you might ask, go through all this trouble with parentheses and such for a language that can only do a few things? Actually, LISP can do a whole lot more than you would expect. Also, it's so simple that it can be written for a personal computer. As an example of a real LISP program, here is one that compares two Lists of any length to see if they are equal.

```
(DEFINE (EQUAL(LAMBDA(X Y)
 (COND ((ATOM X( (COND ((ATOM Y)
                         (EQ X Y)) (T F)))
  ((EQUAL(CAR X) (CAR Y)) (EQUAL(CDR X)
                                (CDR Y)))
  (T F)))) )
```

Believe me, it's not as confusing as it seems. The first line just says that you are defining a function called *EQUAL,* with arguments *X* and *Y*. The next line tests to see if *X* is an atom. If it is, then the program tests to see if *Y* is also an atom. If *Y* is, then the result of the program is set to be the result of (*EQ X Y*)—in other words, it tests to see if the two atoms are equal. (Remember that *EQ* can only test two atoms—never Lists—to see if they are equal.) Anyway, if *Y* is not an atom, then the program returns *F* for false, because *X* is an atom. Since one of the arguments (*X*) is an atom, and the other (*Y*) is not, the two arguments cannot possibly be the same. However, if on the first test (*ATOM X*) was false—if *X* was not an atom—then the program takes the *CAR* of *X* and the *CAR* of *Y,* to see if they are *EQUAL*. In other words, it has to make a recursive call to itself to determine this. If they are *EQUAL,* it then tests the *CDR* parts of *X* and *Y*. If they are *EQUAL,* the result of the program is *T;* if not, it's *F*. Since a List, if it's not just an atom, is made up solely of the *CAR* part and the *CDR* part, if the test succeeds on those parts, the test will succeed on the whole.

Can you write a LISP interpreter? If you understand LISP (which will take a while), you can. There are only a half-dozen different little programs to write. But, believe it or not, LISPing around has produced some of the more sophisticated programs in existence. The famous ELIZA program, which makes a computer seem like a moderately understanding doctor, can be written as a short LISP program. LISP programs have been used to prove statements, manipulate blocks in an imaginary world, and all sorts of other things. ▼



by Daniel Brodsky

Of the many board games man has devised since he first sat playing with bones in his torch-lit cave, the one that perhaps lends itself most naturally to computer competition is Othello. With a sixty-four-square matrix and two-sided (black/white) flippable disks, it would almost seem to have a binary background preconceived with a computer in mind.

Although some Othello programs have been published before, after having been fed this one, your computer should be able to beat those, chips down.

There were certain criteria that I used in writing the program. For one thing, it had to be portable. I wanted a program that could run on almost any machine. That's the reason I chose to code the program in a subset of standard BASIC. The program also had to be easy to modify, for I knew that my initial algorithm was not perfect. Finally, the program had to be a good player. It had to be able to beat an average player.

Most of the code in the program is related to bookkeeping. By that I mean manipulating the board, finding valid moves, and so on. The "heart" of the program is two ten-by-ten arrays called T and D. The T array has the current board positions (0 = blank, 1 = computer, 2 = player). The D array is used as a decision matrix. Each array has the eight-by-eight playing surface in its center.

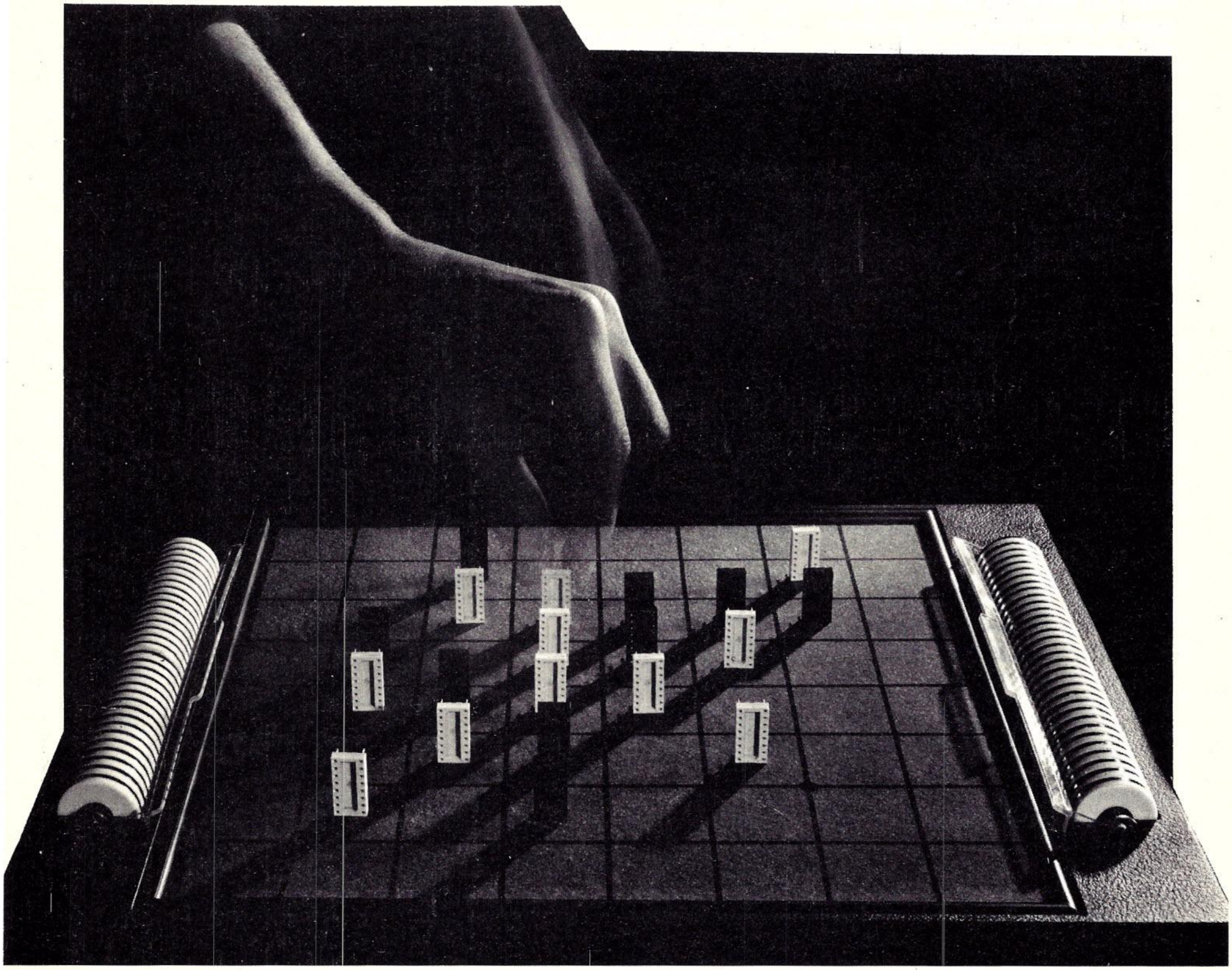
The program uses a positional strategy to determine its move. It does a static (no look-aheads) evaluation of the board and moves to the best location. It assumes that if it is in possession of key locations on the board it will flip over more of its opponent's pieces.

These optimum locations are determined by assigning point values between zero and ten to each location on the decision matrix. The matrix is initialized before each of the program's moves by copying from array W the point values predetermined by the programmer. With each move, the point values in the matrix are updated to correspond to the location of pieces on the board.

The decision matrix is updated in a very simple manner. If the computer is in possession of a corner—for example, T(2,2)—it will change its adjacent locations to 9—D(2,3) and D(3,2)—and the diagonal to 8—D(3,3),

Photograph by Thomas Hall and Tom Hemphill

HELLO●O



D(4,4), D(5,5), D(6,6), and D(7,7). If the computer is in possession of a side location—for example, T(5,2)—it will change that row to 8—D(5,3), D(5,4), D(5,6), and D(5,7). The computer will check each corner and side location, making the changes to each.

The program then chooses its move by finding the location with the highest point value. If there's more than one highest point value, it will pick the move that flips over the most pieces. If this, too, is equal to another move, then it will pick the first one. (I would have made this a random function, but I wanted the play to be consistent.)

This method of play works out fairly well for most moves. There are a few moves that at first caused problems: (1) since the program looked at positions before flips, when it found a move that would flip over all of its opponent's pieces, it might not go there; (2) because of positional choices, it could be tricked into giving up a corner (a key location in the game); (3) there were some cases where a zero-point location was a more advantageous move than a move to a higher-point-value location.

These problems were overcome in two ways. The first way was to add *IF* statements that tested for these cases and then changed the decision matrix. The second way was to set up a "pattern" in array G and to compare that against the board. If the program got a match, it would change the matrix.

The program as it is presently written is a little wasteful in its use of memory (about 11K). By using an extended version of BASIC, the W array could be eliminated, the other arrays shortened, and the number of branches cut down.

Since all variables are integer and there are no character variables, this program should be able to run on any computer that supports BASIC. The only thing that might have to be changed is the heading on the board display (lines 2130, 2280, and 2380). The program was originally written on a PDP 11/40 running under the RSTS operating system. It has also been run on an IBM 370 and a Burroughs B7700.

This Othello program has played over one hundred games on the City University of New York's IBM 370/168 under CALL/OS. It's been winning about 70 percent of the time. ▼

Play it again, Othello

```
OTHELLO2
100        PRINT "OTHELLO VERSION 1.5 —DANIEL BRODSKY"
110 REM
120 REM
130 REM         (C) DANIEL BRODSKY 1977
140 REM
150 REM                  I AM ALLOWING THIS PROGRAM TO BE COPIED OR RUN
160 REM                     BY ANYONE UNDER THE FOLLOWING CONDITIONS:
170 REM
180 REM                  (1) THIS PROGRAM IS NOT TO BE USED FOR COMMERCIAL
190 REM                      PURPOSES WITHOUT THE PERMISSION OF THE AUTHOR.
200 REM
210 REM                  (2) THIS COPYRIGHT NOTICE MUST BE ON EVERY COPY OR
220 REM                      REVISION OF THIS PROGRAM.
230 REM
240 REM
250 REM                        —DANIEL BRODSKY (THE AUTHOR)
260 REM                         138-09 76TH AVE.
270 REM                         FLUSHING NY 11367
280    PRINT
290    PRINT "DO YOU WANT INSTRUCTIONS(1=YES 2=NO)";
300   INPUT R
310   IF R=2 THEN 400
320   PRINT "THIS IS THE GAME OF REVERSI"
330   PRINT
340   PRINT " TO PLAY YOU INPUT THE X,Y COORDINATES OF THE POSITION"
350   PRINT " YOU WANT TO MOVE. IF YOU TYPE 99,9 THE BOARD WILL BE"
360   PRINT " PRINTED "
370   PRINT " X IS DOWN AND Y IS ACROSS"
380   PRINT
390   PRINT "IF YOU TYPE 22,9 ALL YOUR VALID MOVES WILL BE DISPLAYED"
400   PRINT
410   PRINT
420 DIM T(10,10),M(20,4),D(10,10)
430 DIM W(10,10),F(20,2),Z(7,2)
440 DIM G(4,8),O(8)
450 REM ***** INITIALIZE THE DECISION MATRIX
460 GOSUB 1670
470 REM ***** INITIALIZE THE BOARD
480 GOSUB 1930
490 F9 = 0
500 PRINT "WHO GOES FIRST (1=COMPUTER 2=PLAYER) ";
510 INPUT R
520 PRINT "DO YOU WANT THE BOARD DISPLAYED AFTER EACH MOVE (1=Y 2=N)";
530 INPUT Q9
540 PRINT "HOW DO YOU WANT THE BOARD DISPLAYED"
550 PRINT "(1=LONG FORM 2=SHORT FORM) ";
560 INPUT Q8
570 IF Q9=1 THEN 590
580 GOSUB 2100
590 IF R=2 THEN 640
600 T(5,4)=1
610 T(5,5)=1
620 PRINT "COMPUTER MOVES : 4,3"
630 PRINT "                FLIPS   : 4,4"
640 PRINT
650 U1=2
660 U2=1
670 GOSUB 5140
680 IF P2=1 THEN 730
690 PRINT "PLAYER HAS NO VALID MOVES"
700 IF F9 = 1 THEN 1430
710 F9 = 1
```

```
720 GO TO 1080
730 IF Q9=2 THEN 750
740 GOSUB 2100
750 PRINT "PLAYER MOVES
760 INPUT M1,M2
770 IF M1<>99 THEN 800
780 GOSUB 2100
790 GO TO 750
800 IF M1<>22 THEN 890
810 PRINT "PLAYER'S MOVE         NUMBER OF FLIPS"
820 FOR I = 1 TO X STEP 1
830     M1=M(I,1)-1
840     M2=M(I,2)-1
850     PRINT " ";M1;",";M2;"             ";M(I,3)
860     NEXT I
870 PRINT
880 GO TO 750
890 M1=M1+1
900 M2=M2+1
910 F9=0
920 REM ***** CHECK TO SEE IF VALID MOVE
930 REM ***** SET USER1 TO  PLAYER AND USER2 TO COMPUTER
940 U1=2
950 U2=1
960 GOSUB 2500
970 IF P1=1 THEN 1000
980 PRINT "INVALID MOVE . . . TRY AGAIN"
990 GO TO 640
1000 T(M1,M2)=2
1010 FOR I=1 TO L STEP 1
1020     T(F(I,1),F(I,2))=2
1030     F(I,1)=F(I,1)-1
1040     F(I,2)=F(I,2)-1
1050     PRINT "          FLIPS     :";F(I,1);",";F(I,2)
1060     NEXT I
1070 REM ***** DETERMINE IF END OF GAME
1080 GOSUB 2410
1090 IF P1=1 THEN 1430
1100 REM ***** COMPUTE THE DECISION MATRIX
1110 GOSUB 3700
1120 REM ***** FIGURE OUT VALID MOVES
1130 U1=1
1140 U2=2
1150 GOSUB 5140
1160 IF P2=1 THEN 1220
1170 PRINT "COMPUTER HAS NO VALID MOVES"
1180 IF F9=1 THEN 1430
1190 F9=1
1200 GO TO 640
1210 REM ***** FIND OUT BEST MOVE
1220 GOSUB 5390
1230 F9=0
1240 REM ***** FIGURE OUT FLIPS
1250 REM ***** SET USER1 TO COMPUTER USER2 TO PLAYER
1260 U1=1
1270 U2=2
1280 GOSUB 2500
1290 T(M1,M2)=1
1300 M1=M1-1
1310 M2=M2-1
1320 PRINT "COMPUTER MOVES : ";M1;",";M2
1330 FOR I=1 TO L STEP 1
1340     T(F(I,1),F(I,2))=1
1350     F(I,1)=F(I,1)-1
1360     F(I,2)=F(I,2)-1
1370     PRINT "          FLIPS     : ";F(I,1);",";F(I,2)
1380     NEXT I
1390 REM ***** DETERMINE IF END OF GAME
1400 GOSUB 2410
```

```
1410 IF P1=0 THEN 640
1420 REM ***** END OF GAME
1430 C1=0
1440 C2=0
1450 FOR I=2 TO 9 STEP 1
1460     FOR J=2 TO 9 STEP 1
1470         IF T(I,J)=1 THEN 1500
1480         IF T(I,J)=2 THEN 1520
1490         GO TO 1530
1500         C1=C1+1
1510         GO TO 1530
1520         C2=C2+1
1530         NEXT J
1540     NEXT I
1550 PRINT "************ G A M E O V E R **********************"
1560 PRINT
1570 PRINT "COMPUTER HAS ";C1;" PIECES, PLAYER HAS ";C2;" PIECES"
1580 REM ****** PRINT OUT THE BOARD
1590 GOSUB 2100
1600 PRINT "PLAY AGAIN (1=YES 2=NO) ";
1610 INPUT R
1620 IF R=1 THEN 480
1630 GO TO 6400
1640 REM
1650 REM ************ SUBROUTINES *************
1660 REM
1670 REM ***** INITIALIZE DECISION MATRIX
1680 FOR I=1 TO 10 STEP 1
1690     FOR J=1 TO 10 STEP 1
1700         READ W(I,J)
1710         NEXT J
1720     NEXT I
1730 FOR I=1 TO 4 STEP 1
1740  FOR J=1 TO 8 STEP 1
1750     READ G(I,J)
1760     NEXT J
1770  NEXT I
1780 RETURN
1790 DATA 00,00,00,00,00,00,00,00,00,00
1800 DATA 00,10,00,09,09,09,09,00,10,00
1810 DATA 00,00,00,09,05,05,09,00,00,00
1820 DATA 00,09,09,07,07,07,07,09,09,00
1830 DATA 00,09,05,07,06,06,07,05,09,00
1840 DATA 00,09,05,07,06,06,07,05,09,00
1850 DATA 00,09,09,07,07,07,07,09,09,00
1860 DATA 00,00,00,09,05,05,09,00,00,00
1870 DATA 00,10,00,09,09,09,09,00,10,00
1880 DATA 00,00,00,00,00,00,00,00,00,00
1890 DATA 00,00,02,02,02,02,01,03
1900 DATA 00,00,02,02,02,01,03,03
1910 DATA 00,00,02,02,01,01,03,03
1920 DATA 00,00,02,01,01,01,03,03
1930 REM ***** SET UP THE BOARD
1940 FOR I=2 TO 9 STEP 1
1950     FOR J=2 TO 9 STEP 1
1960         T(I,J)=0
1970         NEXT J
1980     NEXT I
1990 FOR I=1 TO 10 STEP 1
2000     T(I,1)=9
2010     T(I,10)=9
2020     T(1,I)=9
2030     T(10,I)=9
2040     NEXT I
2050 T(5,5)=2
2060 T(6,6)=2
2070 T(6,5)=1
2080 T(5,6)=1
2090 RETURN
```

```
2100 REM ***** PRINT OUT THE BOARD
2110 IF Q8=1 THEN 2230
2120  PRINT " 0=BLANK, 1=COMPUTER, 2=PLAYER"
2130  PRINT "    1 2 3 4 5 6 7 8"
2140 FOR I=2 TO 9 STEP 1
2150       I1=I-1
2160       PRINT I1;" ";
2170  FOR J=2 TO 9 STEP 1
2180   PRINT T(I,J);
2190   NEXT J
2200  PRINT
2210  NEXT I
2220 RETURN
2230 PRINT
2240 PRINT "              CURRENT BOARD"
2250 PRINT
2260 PRINT " 0 = BLANK    1 = COMPUTER    2 = PLAYER"
2270 PRINT
2280  PRINT "         (1) (2) (3) (4) (5) (6) (7) (8)"
2290 FOR I=2 TO 9 STEP 1
2300       I1=I-1
2310       PRINT "  (";I1;")  ";
2320       FOR J=2 TO 9 STEP 1
2330          PRINT T(I,J);" ";
2340           NEXT J
2350       PRINT "  (";I1;")"
2360          PRINT
2370       NEXT I
2380  PRINT "         (1) (2) (3) (4) (5) (6) (7) (8)"
2390 PRINT
2400 RETURN
2410 REM ***** IS IT THE END OF THE GAME ????
2420 P1=0
2430 FOR I=2 TO 9 STEP 1
2440        FOR J=2 TO 9 STEP 1
2450        IF T(I,J) = 0 THEN 2490
2460        NEXT J
2470        NEXT I
2480 P1=1
2490 RETURN
2500 REM ***** IS IT A VALID MOVE ??
2510 P1=0
2520 L=0
2530 IF M1<2 THEN 2560
2540 IF M2<2 THEN 2560
2550 GO TO 2580
2560 P1=0
2570 RETURN
2580 IF M1>9 THEN 2610
2590 IF M2>9 THEN 2610
2600 GO TO 2630
2610 P1=0
2620 RETURN
2630 IF T(M1,M2) = 0 THEN 2660
2640 P1=0
2650 RETURN
2660 IF T(M1+1,M2)<>U2 THEN 2770
2670 Q=0
2680 FOR I=M1+1 TO 9 STEP 1
2690        IF T(I,M2)=U1 THEN 2760
2700        IF T(I,M2)=0 THEN 2770
2710        Q=Q+1
2720        Z(Q,1)=I
2730        Z(Q,2)=M2
```

Y →

X ↓	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	10	0	9	9	9	9	0	10	0
3	0	0	0	9	5	5	9	0	0	0
4	0	9	9	7	7	7	7	9	9	0
5	0	9	5	7	6	6	7	5	9	0
6	0	9	5	7	6	6	7	5	9	0
7	0	9	9	7	7	7	7	9	9	0
8	0	0	0	9	5	5	9	0	0	0
9	0	10	0	9	9	9	9	0	10	0
10	0	0	0	0	0	0	0	0	0	0

INITIAL VALUE OF DECISION MATRIX (Array D)

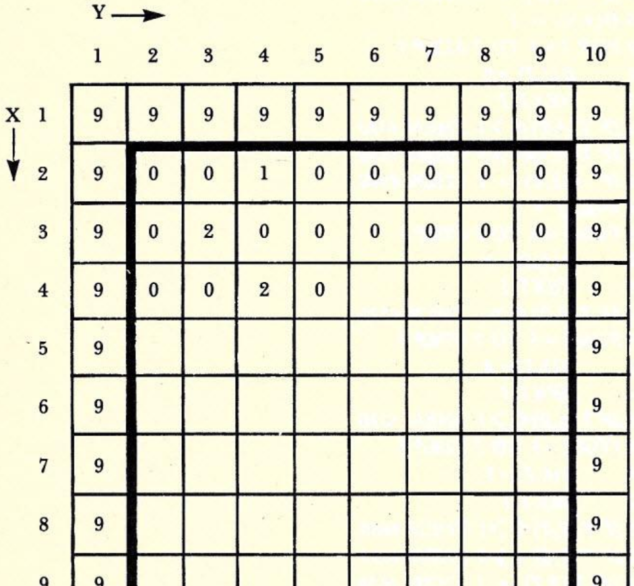
Y →

X ↓	1	2	3	4	5	6	7	8	9	10
1	9	9	9	9	9	9	9	9	9	9
2	9	0	0	0	0	0	0	0	0	9
3	9	0	0	1	2	2	0	0	0	9
4	9	0	0	0	1	1	0	0	0	9
5	9	0	0	0	0	1	0	0	0	9
6	9	0	0	0	0	0	0	0	0	9
7	9	0	0	0	0	0	0	0	0	9
8	9	0	0	0	0	0	0	0	0	9
9	9	0	0	0	0	0	0	0	0	9
10	9	9	9	9	9	9	9	9	9	9

PROBLEM 1: THE COMPUTER BYPASSING A WIN
(Array T)

In this condition the computer will go to location T(2,5) or T(2,6). If it were to go to location T(3,7), it would flip over all the player's pieces and win the game.

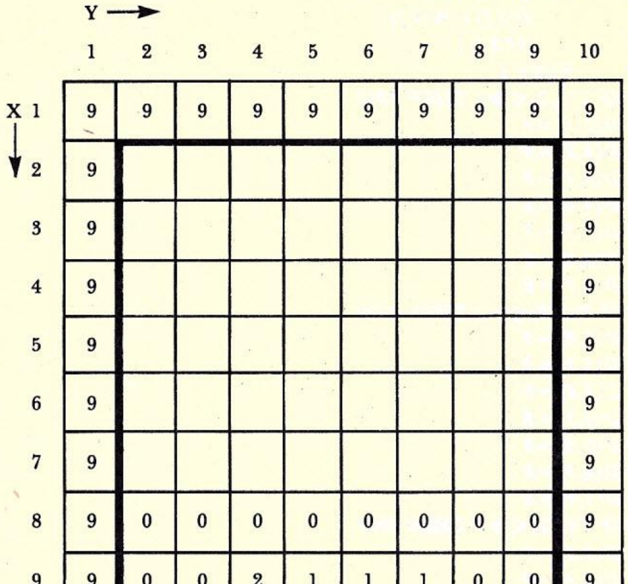
```
2740     NEXT I
2750 GO TO 2770
2760 GOSUB 3630
2770 IF T(M1+1,M2+1)< >U2 THEN 2890
2780 Q=0
2790 FOR I=1 TO 7 STEP 1
2800     IF T(M1+I,M2+I)  =9 THEN 2890
2810     IF T(M1+I,M2+I)  = 0 THEN 2890
2820     IF T(M1+I,M2+I)  = U1 THEN 2880
2830     Q=Q+1
2840     Z(Q,1) =M1+I
2850     Z(Q,2) =M2+I
2860     NEXT I
2870 GO TO 2890
2880 GOSUB 3630
2890 IF T(M1,M2+1)< >U2 THEN 3000
2900 Q=0
2910 FOR I=M2+1 TO 9 STEP 1
2920     IF T(M1,I) =U1 THEN 2990
2930     IF T(M1,I) =0 THEN 3000
2940     Q=Q+1
2950     Z(Q,1) =M1
2960     Z(Q,2) =I
2970     NEXT I
2980 GO TO 3000
2990 GOSUB 3630
3000 IF T(M1-1,M2-1)< >U2 THEN 3120
3010 Q=0
3020 FOR I=1 TO 7 STEP 1
3030     IF T(M1-I,M2-I)  =9 THEN 3120
3040     IF T(M1-I,M2-I)  = 0 THEN 3120
3050     IF T(M1-I,M2-I)  = U1 THEN 3110
3060     Q=Q+1
3070     Z(Q,1) =M1-I
3080     Z(Q,2) =M2-I
3090     NEXT I
3100 GO TO 3120
3110 GOSUB 3630
3120 IF T(M1-1,M2)< >U2 THEN 3240
3130 Q=0
3140 FOR I=1 TO 7 STEP 1
3150     IF T(M1-I,M2)  = 9 THEN 3240
3160     IF T(M1-I,M2)  = 0 THEN 3240
3170     IF T(M1-I,M2)  = U1 THEN 3230
3180     Q=Q+1
3190     Z(I,1) =M1-I
3200     Z(I,2) =M2
3210     NEXT I
3220 GO TO 3240
3230 GOSUB 3630
3240 IF T(M1-1,M2+1)< >U2 THEN 3360
3250 Q=0
3260 FOR I=1 TO 7 STEP 1
3270     IF T(M1-I,M2+I)  = 9 THEN 3360
3280     IF T(M1-I,M2+I)  = 0 THEN 3360
3290     IF T(M1-I,M2+I)  =U1 THEN 3350
3300     Q=Q+1
3310     Z(Q,1) =M1-I
3320     Z(Q,2) =M2+I
3330     NEXT I
3340 GO TO 3360
3350 GOSUB 3630
3360 IF T(M1,M2-1)< >U2 THEN 3480
3370 Q=0
```



	1	2	3	4	5	6	7	8	9	10
1	9	9	9	9	9	9	9	9	9	9
2	9	0	0	1	0	0	0	0	0	9
3	9	0	2	0	0	0	0	0	0	9
4	9	0	0	2	0					9
5	9									9
6	9									9
7	9									9
8	9									9
9	9									9
10	9	9	9	9	9	9	9	9	9	9

PROBLEM 2: TRICKING THE COMPUTER INTO GIVING AWAY A CORNER
(Array T)

In the condition given above, the computer will go to location T(4,2) and flip T(3,3), since it will go for a location which has a point value of 9 on the D matrix. The player will then go to location T(2,2), thereby winning the corner.



	1	2	3	4	5	6	7	8	9	10
1	9	9	9	9	9	9	9	9	9	9
2	9									9
3	9									9
4	9									9
5	9									9
6	9									9
7	9									9
8	9	0	0	0	0	0	0	0	0	9
9	9	0	0	2	1	1	1	0	0	9
10	9	9	9	9	9	9	9	9	9	9

PROBLEM 3: GOING TO A ZERO LOCATION ON THE DECISION MATRIX
(Array T)

In this case it is all right for the computer to go to location T(9,3), which is a zero location on the D matrix, since the only way to flip the computer's pieces is for the player to gain possession of one of the corners.

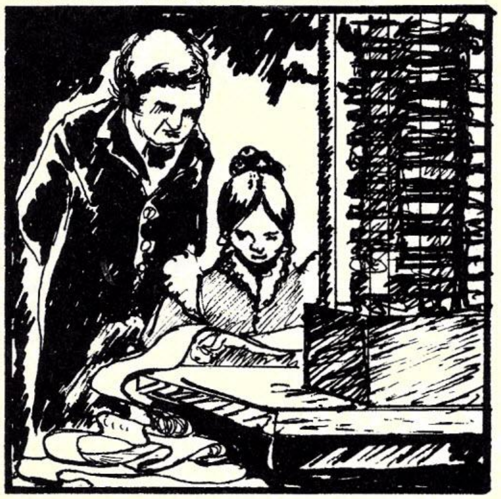
```
3380 FOR I=1 TO 7 STEP 1
3390       IF T(M1,M2-I) =9 THEN 3480
3400       IF T(M1,M2-I) =0 THEN 3480
3410       IF T(M1,M2-I) = U1 THEN 3470
3420       Q=Q+1
3430       Z(Q,1) =M1
3440       Z(Q,2) =M2-I
3450       NEXT I
3460 GO TO 3480
3470 GOSUB 3630
3480 IF T(M1+1,M2-1)< >U2 THEN 3600
3490 Q=0
3500 FOR I=1TO 7 STEP 1
3510       IF T(M1+I,M2-I) =9 THEN 3600
3520       IF T(M1+I,M2-I) = 0 THEN 3600
3530       IF T(M1+I,M2-I) = U1 THEN 3590
3540       Q=Q+1
3550       Z(Q,1) =M1+I
3560       Z(Q,2) =M2-I
3570       NEXT I
3580 GO TO 3600
3590 GOSUB 3630
3600 IF L = 0 THEN 3620
3610 P1=1
3620 RETURN
3630 REM ***** SHIFT ROUTINE
3640 FOR Q2=1 TO Q STEP 1
3650       L=L+1
3660       F(L,1) =Z(Q2,1)
3670       F(L,2) =Z(Q2,2)
3680       NEXT Q2
3690 RETURN
3700 REM ***** COMPUTE THE DECISION MATRIX
3710 FOR I=1 TO 10
3720       FOR J=1 TO 10
3730             D(I,J) = W(I,J)
3740             NEXT J
3750       NEXT I
3760 IF T(2,2)< >1 THEN 3840
3770 D(2,3) =9
3780 D(3,2) =9
3790 D(3,3) =8
3800 D(4,4) =8
3810 D(5,5) =8
3820 D(6,6) =8
3830 D(7,7) =8
3840 IF T(2,9)< >1 THEN 3920
3850 D(2,8) =9
3860 D(3,9) =9
3870 D(3,8) =8
3880 D(4,7) =8
3890 D(5,6) =8
3900 D(6,5) =8
3910 D(7,4) =8
3920 IF T(9,2)< >1 THEN 4000
3930 D(8,2) =9
3940 D(9,3) =9
3950 D(8,3) =8
3960 D(7,4) =8
3970 D(6,5) =8
3980 D(5,6) =8
3990 D(4,7) =8
4000 IF T(9,9)< >1 THEN 4080
4010 D(9,8) =9
4020 D(8,9) =9
4030 D(8,8) =8
4040 D(7,7) =8
4050 D(6,6) =8
4060 D(5,5) =8
```

```
4070 D(4,4) =8
4080 IF T(2,2)< > 2 THEN 4120
4090 D(2,3) =1
4100 D(3,2) =1
4110 D(3,3) =1
4120 IF T(2,9)< > 2 THEN 4160
4130 D(2,8) =1
4140 D(3,8) =1
4150 D(3,9) =1
4160 IF T(9,9)< > 2 THEN 4200
4170 D(8,9) =1
4180 D(8,7) =1
4190 D(9,8) =1
4200 IF T(9,2)< >2 THEN 4240
4210 D(8,2) =1
4220 D(8,3) =1
4230 D(9,3) =1
4240 IF T(2,4)< >1 THEN 4310
4250 IF T(3,3) < >2 THEN 4280
4260 IF T(2,2)  = 1 THEN 4280
4270 D(4,2)  = 1
4280 FOR I=3 TO 7 STEP 1
4290       D(I,4) =8
4300       NEXT I
4310 IF T(2,5)< >1 THEN 4350
4320 FOR I=3 TO 7 STEP 1
4330       D(I,5) =8
4340       NEXT I
4350 IF T(2,6) < >1 THEN 4390
4360 FOR I=3 TO 7 STEP 1
4370       D(I,6) =8
4380       NEXT I
4390 IF T(2,7)< >1 THEN 4460
4400 IF T(3,8)< >2 THEN 4430
4410 IF  T(2,9)  =  1 THEN 4430
4420 D(4,9)  =  1
4430 FOR I=3 TO 7 STEP 1
4440       D(I,7) =8
4450       NEXT I
4460 IF T(4,2)< >1 THEN 4530
4470 IF T(3,3)< >2 THEN 4500
4480 IF  T(2,2)  =  1 THEN 4500
4490 D(2,4)  =  1
4500 FOR I=3 TO 7 STEP 1
4510       D(4,I) =8
4520       NEXT I
4530 IF T(5,2)< >1 THEN 4570
4540 FOR I=3 TO 7 STEP 1
4550       D(5,I) =8
4560       NEXT I
4570 IF T(6,2)< >1 THEN 4610
4580 FOR I=3 TO 7 STEP 1
4590       D(6,I) =8
4600       NEXT I
4610 IF T(7,2)< >1 THEN 4680
4620 IF T(8,3) < >2 THEN 4650
4630 IF  T(9,2)  =  1 THEN 4650
4640 D(9,4)  =  1
4650 FOR I=3 TO 7 STEP 1
4660       D(7,I) =8
4670       NEXT I
4680 IF T(9,4)< >1 THEN 4750
4690 IF T(8,3)< >2 THEN 4720
4700 IF  T(9,2)  =  1 THEN 4720
4710 D(7,2)  =  1
4720 FOR I=1 TO 5 STEP 1
4730       D(9-I,4) =8
4740       NEXT I
4750 IF T(9,5)< >1 THEN 4790
```

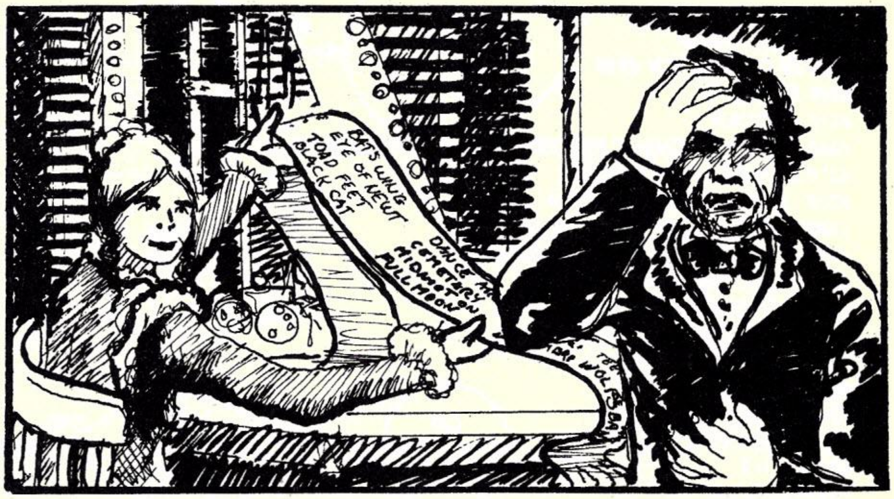
```
4760 FOR I=1 TO 5 STEP 1
4770      D(9-I,5)=8
4780      NEXT I
4790 IF T(9,6)<>1 THEN 4830
4800 FOR I=1 TO 5 STEP 1
4810      D(9-I,6)=8
4820      NEXT I
4830 IF T(9,7)<>1 THEN 4900
4840 IF T(8,8)<>2 THEN 4870
4850 IF  T(9,9)  =  1 THEN 4870
4860 D(7,9)  =  1
4870 FOR I=1 TO 5 STEP 1
4880      D(9-I,7)=8
4890      NEXT I
4900 IF T(7,9)<>1 THEN 4970
4910 IF T(8,8)<>2 THEN 4940
4920 IF  T(9,9)  =  1 THEN 4940
4930 D(9,7)=1
4940 FOR I=1 TO 5 STEP 1
4950      D(7,9-I)=8
4960      NEXT I
4970 IF T(6,9)<>1 THEN 5010
4980 FOR I=1 TO 5 STEP 1
4990      D(6,9-I)=8
5000      NEXT I
5010 IF T(5,9)<>1 THEN 5050
5020 FOR I=1 TO 5 STEP 1
5030      D(5,9-I)=8
5040      NEXT I
5050 IF T(4,9)<>1 THEN 5120
5060 IF T(3,3)<>2 THEN 5090
5070 IF  T(2,9)  =  1 THEN 5090
5080 D(2,7)  =  1
5090 FOR I=1 TO 5 STEP 1
5100      D(4,9-I)=8
5110      NEXT I
5120 GOSUB 5770
5130 RETURN
5140 REM ***** FIGURE OUT VALID MOVES ...COMPUTER
5150 X=0
5160 P2=0
5170 REM ***** FIND OUT HOW MANY OF PLAYERS PIECES ON BOARD
5180 Q7=0
5190 FOR M1=2 TO 9 STEP 1
5200  FOR M2=2 TO 9 STEP 1
5210    IF T(M1,M2)<>2 THEN 5230
5220    Q7=Q7+1
5230    NEXT M2
```

```
5240  NEXT M1
5250 FOR M1=2 TO 9 STEP 1
5260      FOR M2=2 TO 9 STEP 1
5270           GOSUB 2500
5280           IF P1=0 THEN 5360
5290           P2=1
5300           X=X+1
5310           M(X,1)=M1
5320           M(X,2)=M2
5330           M(X,3)=L
5340      IF L<>Q7 THEN 5360
5350      D(M1,M2)=11
5360           NEXT M2
5370      NEXT M1
5380 RETURN
5390 REM ***** FIND THE BEST MOVE
5400 FOR I=1 TO X STEP 1
5410      M(I,4)=D(M(I,1),M(I,2))
5420      NEXT I
5430 K2=4
5440 GOSUB 5570
5450 IF X=1 THEN 5540
5460 IF M(1,4)<>M(2,4) THEN 5540
5470 FOR I=1 TO X-1 STEP 1
5480      IF M(I,4)<>M(I+1,4) THEN 5510
5490      NEXT I
5500 GO TO 5520
5510 X=I
5520 K2=3
5530 GOSUB 5570
5540 M1=M(1,1)
5550 M2=M(1,2)
5560 RETURN
5570 REM ***** SORT MOVES
5580 F=1
5590 FOR J=1 TO X STEP 1
5600      IF F=0 THEN 5760
5610      F=0
5620      FOR I=1 TO X-1 STEP 1
5630           IF M(I,K2)>M(I+1,K2) THEN 5740
5640           F=1
5650           FOR K=1 TO 4
5660                V(K)=M(I,K)
5670                NEXT K
5680           FOR K=1 TO 4
5690                M(I,K)=M(I+1,K)
5700                NEXT K
5710           FOR K=1 TO 4
```

BABBAGE AND LOVELACE



"Lovelace dear, what is *it?"*

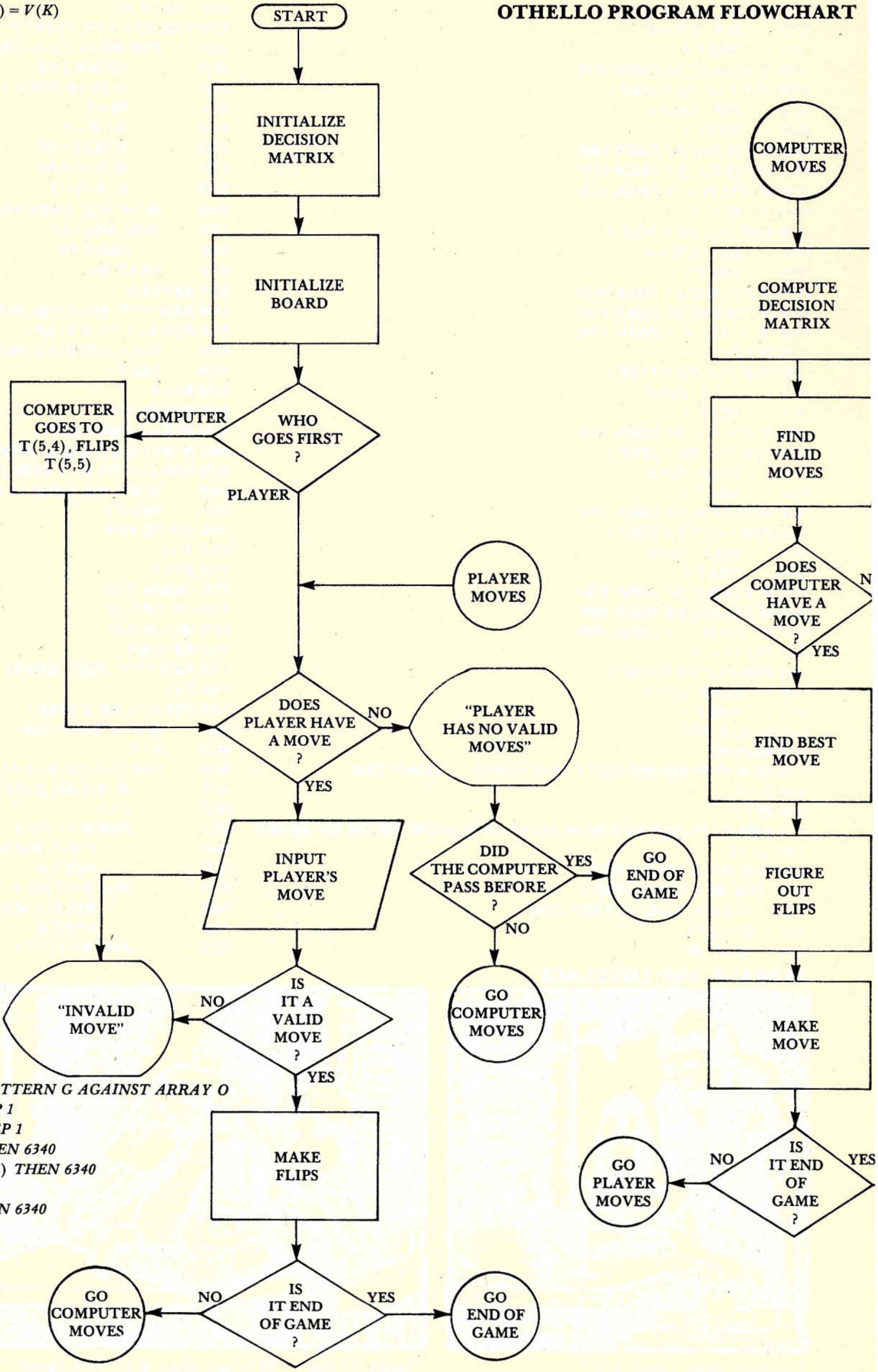


"Don't be thick, Babbage dear, it's a hex dump."

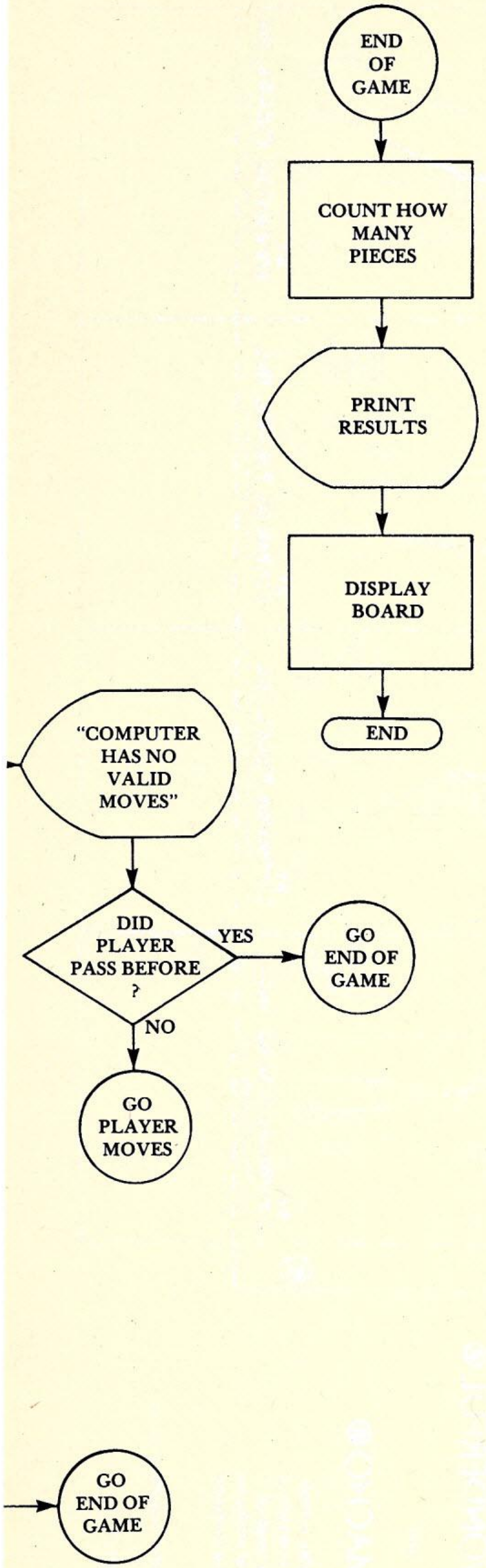
Cartoon by Steve Gerling, story line by Andrew Singer

```
5720              M(I+1,K) = V(K)
5730              NEXT K
5740          NEXT I
5750      NEXT J
5760 RETURN
5770 FOR I=1 TO 8 STEP 1
5780   O(I) = T(2,I+1)
5790 NEXT I
5800 GOSUB 6270
5810 IF O1=1 THEN 5830
5820 D(2,3) = 9
5830 FOR I=1 TO 8 STEP 1
5840   O(I) = T(2,10-I)
5850 NEXT I
5860 GOSUB 6270
5870 IF O1=1 THEN 5890
5880 D(2,8) = 9
5890 FOR I=1 TO 8 STEP 1
5900   O(I) = T(I+1,2)
5910 NEXT I
5920 GOSUB 6270
5930 IF O1=1 THEN 5950
5940 D(3,2) = 9
5950 FOR I=1 TO 8 STEP 1
5960   O(I) = T(10-I,2)
5970 NEXT I
5980 GOSUB 6270
5990 IF O1=1 THEN 6010
6000 D(8,2) = 9
6010 FOR I=1 TO 8 STEP 1
6020   O(I) = T(9,I+1)
6030 NEXT I
6040 GOSUB 6270
6050 IF O1=1 THEN 6070
6060 D(9,3) = 9
6070 FOR I=1 TO 8 STEP 1
6080   O(I) = T(9,10-I)
6090 NEXT I
6100 GOSUB 6270
6110 IF O1=1 THEN 6130
6120 D(9,8) = 9
6130 FOR I=1 TO 8 STEP 1
6140   O(I) = T(I+1,9)
6150 NEXT I
6160 GOSUB 6270
6170 IF O1=1 THEN 6190
6180 D(3,9) = 9
6190 FOR I=1 TO 8 STEP 1
6200   O(I) = T(10-I,9)
6210 NEXT I
6220 GOSUB 6270
6230 IF O1=1 THEN 6250
6240 D(8,9) = 9
6250 RETURN
6260 REM ***** CHECK PATTERN G AGAINST ARRAY O
6270 FOR O2=1 TO 4 STEP 1
6280   FOR O3=1 TO 8 STEP 1
6290     IF G(O2,O3) = 3 THEN 6340
6300     IF G(O2,O3) = O(O3)  THEN 6340
6310     GO TO 6370
6320     IF O(O3)< >2 THEN 6340
6330     GO TO 6370
6340     NEXT O3
6350   O1=0
6360   RETURN
5370 NEXT O2
6380 O1=1
6390 RETURN
6400 END
```

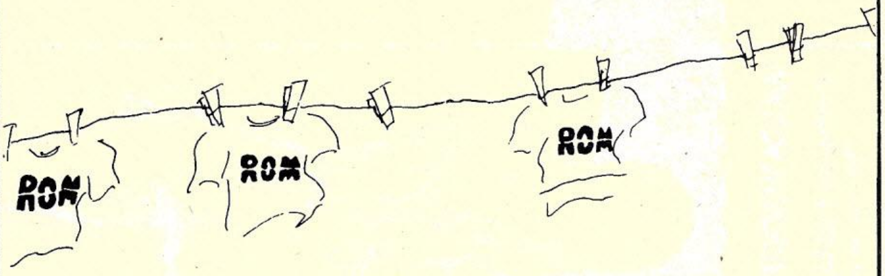
OTHELLO PROGRAM FLOWCHART



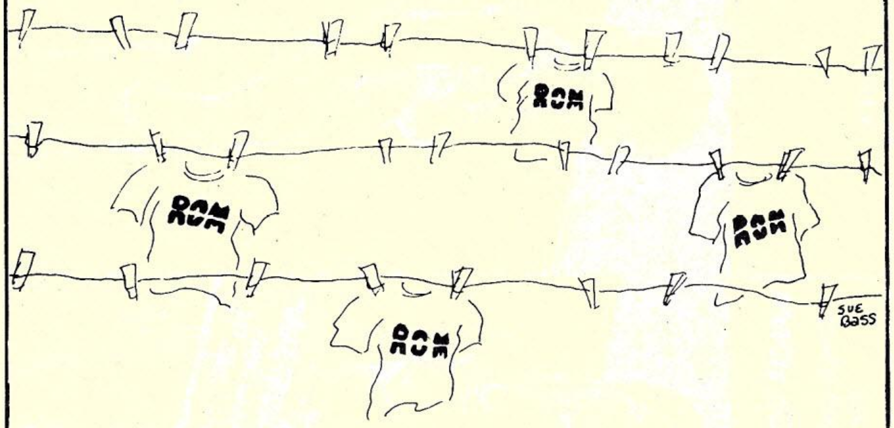
START
INITIALIZE DECISION MATRIX
INITIALIZE BOARD
WHO GOES FIRST ?
COMPUTER
PLAYER
COMPUTER GOES TO T(5,4), FLIPS T(5,5)
PLAYER MOVES
DOES PLAYER HAVE A MOVE ?
NO
YES
"PLAYER HAS NO VALID MOVES"
DID THE COMPUTER PASS BEFORE ?
YES
NO
GO END OF GAME
GO COMPUTER MOVES
INPUT PLAYER'S MOVE
IS IT A VALID MOVE ?
NO
YES
"INVALID MOVE"
MAKE FLIPS
IS IT END OF GAME ?
NO
YES
GO COMPUTER MOVES
GO END OF GAME
COMPUTER MOVES
COMPUTE DECISION MATRIX
FIND VALID MOVES
DOES COMPUTER HAVE A MOVE ?
N
YES
FIND BEST MOVE
FIGURE OUT FLIPS
MAKE MOVE
IS IT END OF GAME ?
NO
YES
GO PLAYER MOVES


END OF GAME
COUNT HOW MANY PIECES
PRINT RESULTS
DISPLAY BOARD
END
"COMPUTER HAS NO VALID MOVES"
DID PLAYER PASS BEFORE ?
YES
NO
GO END OF GAME
GO PLAYER MOVES
GO END OF GAME

HANG OUT WITH THE BEST!!!



Get your ROM T-shirt now!


ROM
ROM
ROM
ROM
SUE BASS

ROM
COMPUTER APPLICATIONS FOR LIVING

Order from:
ROM Publications Corp.
Route 97
Hampton, CT 06247

☐S Qty.______ ☐Child S Qty.______
☐M Qty.______ ☐Child L Qty.______
☐L Qty.______
☐XL Qty.______
☐Check or money order enclosed. $5 ppd.
☐Master Charge ☐BankAmericard 2 for $9 ppd.
Exp. date Card#

Please ship to:
Name____________________________
Address____________________________
City______________ State________ Zip________

Please allow 4 to 6 weeks for delivery.

All right, Gang—

ROM Publications Corp. and Companion Pieces, Inc., proudly present a genuine cut-and-fold scale model of the package for Companion Pieces, Inc.'s WANDA WONDERFUL®, the life-size, computerized, robotized adult playmate! The ultimate pleasure!! Makes traditional relationships obsolete!!!

Directions:

Simply cut out on heavy solid outlines and fold on dotted lines.

Narrow tab A goes under far edge and is held in place with tape, glue, or whatever.

Then fold or tape top and bottom flaps as logic dictates, and there you have it!

Watch for MAX MATCHO® and other models in later advertisements!

Start your collection now!

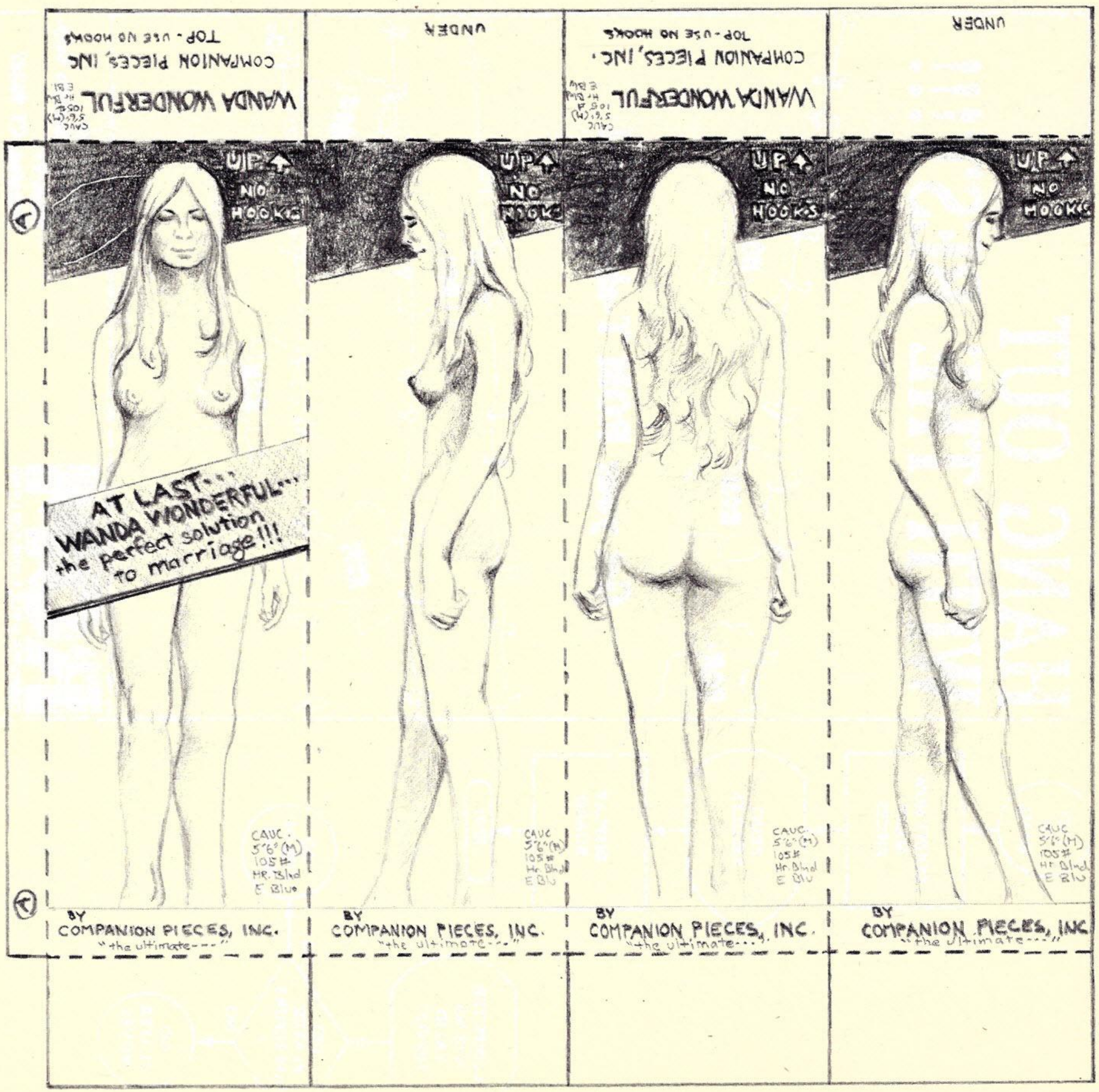
WANDA WONDERFUL®

and

MAX MACHO®

available in finer
stores everywhere
in a range of
physical attributes
to please everyone!

for catalogues and further literature, write:
Companion Pieces, Inc.
4515 Barleycorn Avenue
Los Angeles, CA


WANDA WONDERFUL
COMPANION PIECES, INC
TOP - USE NO HOOKS
UNDER
UP
NO HOOKS
A
AT LAST... WANDA WONDERFUL... the perfect solution to marriage!!!
CAUC. 5'6" (M) 105# HR. Blnd E Blue
BY COMPANION PIECES, INC.
"the ultimate---"

Illustrated by Steve Gerling

The Oedipus Transaction

A Short Story by Robert Abel

J. D. "Steady" Stedmund could hardly find time to grab a lunch anymore because he had come up up with a new investment scheme that required him to be at the computer console every second. Rosa, his executive assistant who was in the computer loft with him, jokingly suggested that Steady take his meals intravenously, and was a little taken aback when he stopped for a moment—a rare thing for Steady!—to apparently give the idea serious thought. He boomed, "You think that's funny, don't you? But do you realize how much it costs me to have lunch? About $100 a minute. That'd buy a whole lot of Peking Duck, Quiche Lorraine, or Himalayan spumante, you understand?"

The investment scheme was actually a spinoff from Steady's main business, Companion Pieces, Inc., a concern which manufactured life-sized, roboticized "guys and dolls" with as many companionable human attributes as the research and development division could simulate, create, and program into the assembly process. The firm was not doing well. In order to set back competition, Steady had found it imperative to expand his operation into other commercial zones, and continually pare down his profit margin. Although he was making a fair profit overall, each individual enterprise was pretty shaky.

"I'm running like crazy to stand still," he told Rosa. "It's a deck of cards. If Play Dolls gets any more of the East Coast market, if people don't go for the Companion Pieces fashions we've got coming out in the fall, if they don't rent our cars or use our laundromats. . . ."

"If, if, if," Rosa said. "Keep your cool, and you'll be all right. Hey! Why do they call you 'Steady' after all?"

"Yeah. Why do they?" he replied.

What it had come down to was this: Steady was now investing out of his cash flow, any money that was *temporarily suspended* by any accounting transaction. Between the time a check was deposited and the time a check cleared, the money represented by the check was in a kind of financial limbo. In precomputer days, the money could not have been touched. But now, thanks to the computer's fantastically speedy calculating abilities, this money could be "loaned" to others or "borrowed" by Steady himself for a few days, a few hours, or a few minutes—even seconds. Instead of going from point A to point B directly, the money went from A to C, D, E, or F before it arrived at B. This required, first of all, a computer program which would keep track of all financial transactions in the Companion Pieces conglomerate at all times, and defer all payouts of any kind to the last possible second. Steady had at his disposal a "Limbo Money Accounting Record," or LIMAR, as Rosa dubbed it.

"What's the LIMAR?" Steady would shout.

"$60,018.20. No, just dropped to $35,212," Rosa would reply, referring to the amount of money suspended by accounting transactions at that second.

Loans were based on the gamble that the normal amount of cash in the system was about $15,036 a minute.

"Tell me if it hits $100,000," Steady might say. "If it doesn't get there by 10:30, instruct the West Coast offices to pay their shopworkers a day early." Such transactions would put a considerable amount of money in limbo, since many of the checks would not be automatically deposited until late the following day. Of course, company policy encouraged employees to select "automatic deposit" for their paychecks so that the amount of limbo money available could be more freely manipulated.

The scheme also required that Steady work most furiously when most of the money was in limbo. This meant, in turn, a West Coast office which was "behind time"—open for as much as five hours after the East Coast branch offices had closed. This gave Steady a full five hours of staff time at normal salary to devote to investing his money-in-limbo. He also worked late into the night and on weekends.

Naturally, none of this furious financial footwork would have had much meaning without a steady demand for short-term loans and a willingness to pay interest on them. To some extent, the loan period could be stretched in the case of single large-scale transactions. These loans were based on the gamble that the normal amount of cash between records in the Companion Pieces system was about $15,036 a minute. If the rate threatened to fall below that level (or, as Rosa put it, "get a hole in it"), then Steady would have to create a transaction—buy some of his own goods out of profits, or sell some of his own stock and buy it back again. To "plug the hole," or make these deals to keep the limbo-money level up enough to support a relatively long-term loan, Steady had created his own private investment corporation. The Companion Corporation had no intention of keeping the goods it bought from itself, and obviously Steady could not sell his stocks without a clear guarantee they would instantly revert back to him. The private investment corporation was run by Steady's daughter.

Beyond these "long-timers," or LTs, Steady was more and more being forced to rely on FTs, "Futures Transactions," perfectly-timed loans for strictly limited periods of time. At first, these turned out to be extremely risky because any transaction that lapsed over the time boundaries could so easily develop a hole, without the money needed to complete it. To prevent this, Steady had carefully built up a reticulae of busywork transactions in the conglomerate that would provide a steady pool of limbo funds which could be used to couch an FT that did not—as usually was the case—go smoothly.

"We've got an FT scheduled at 10:04:06 to 10:15 A.M. today," Rosa would say.

"For how much?" Steady asked.

"$10,540."

"Who's using it?"

"Geronimo Construction."

"We'd better jack up the pool then," Steady would say. "Those guys are never on time. We've got ten minutes to get $10,540 for another half hour. What's available for limbo, Rosa?"

"How about repairs to the fleet? They're due to be paid off."

"No good. I'm using them to cover an FT from Connalt Industries at 3:30. What else?"

"Television ads in Toledo. That would be a partial."

"Right! Good. A small-town TV station can't be that up-to-date on their accounting."

"Good thing for us," Rosa said.

"Right again! Why do you think I buy off all the great programmers? If and when everybody gets the same degree of computer sophistication, we've had it with this operation. When they're as fast as we are, no more something for nothing."

"That's a long way off, isn't it?"

"Pray for an end to those technological breakthroughs, Rosa. They make computers any faster, our goose is cooked."

"I always did like goose," Rosa said. "In orange sauce. Wild rice on the side."

Steady groaned. Geronimo was late on their payback. He sold three vans on the East Coast to a West Coast office to keep the LIMAR up, and a second later had to sell them back again. Limbo money. There was millions of it—evanescent, unrecorded, outlaw money—but millions simply floating between payment and collection. You had to be plenty fast to grab it and use it and put it back again, but what was a computer good for if not this kind of financial sleight of hand? And plenty of customers so far. No need to worry about that. Was there?

What Steady had hoped for in a daughter was a beautiful heartbreaker, but what he got instead was a magnate.

What Steady Stedmund had hoped for in a daughter was a beautiful little heartbreaker, but what he got instead was a magnate. It was just as well. She did him immense credit by succeeding handsomely in three separate businesses before she was thirty. The first was "Home Sweetners," appliances that recycled human wastes into methane and humus. Many experts had considered such devices as beyond the cultural pale of most Americans, but Julia blew the market wide open with a peppy advertising campaign that made "Home Sweetners" the most "in" thing in the country. Talk about waste-conversion processes became the *sine qua non* of the kaffee klatch and cocktail party.

This triumph was followed by Julia's successful entry into "Medicomp," a computer system which would dispense appropriate medicines, advice, and warnings according to the symptoms coded in by the terminal's users. Medicomp offered its own medicines to the system's subscribers at a discount, along with various health-insurance packages, and the "Don't Forget" program which reminded patients when to take their medicines, to exercise, to rest, and which generally maintained their health lives. ("It's 2:00, Mrs. Reynolds. Time to take your pink pill and a glass of water. Be sure to have a salad with your fish tonight.")

As if this were not enough, at the tender age of 27, Julia began to market "Playback!" This was a videotape system coupled to a computer. In an astonishingly short time, due to Julia's marketing ingenuity, "Playback!" became to the 1990s what CB radios had been in her childhood years. By using the relatively common information-storage-and-retrieval systems, Julia provided a service (at a small cost) by which people could store videotapes they made of themselves on a central computer disk. These could then be made available to as limited or as broad an audience as they wished. For another small fee, a person could subscribe to any number of "Playback!" series, and could access them on their own video consoles. The series ranged from political views to cross-country teen talk to adult fun. But, in general, all the series were pretty uninhibited, and Steady himself—in the days when he had any time for it—had enjoyed accessing his daughter's system to hear Louis Thornton of Ames, Iowa denounce the Dairyfarmers' Association for lackadasical lobbying, watch "Ginger Snap" (an obvious PB handle) do an impromptu bump and grind in front of her chartreuse refrigerator, or study the way Han Yu Ling from San Francisco made wonderful, paper-thin Manderin pancakes in his wok. The television industry practically crumbled.

New Fortune magazine described Julia Stedmund as "the Seventh Wonder of the New Age business world" and devoted eighteen pages to an interview with her. When *New Fortune* had interviewed Steady, five years previously, they devoted an unprecedented six pages of copy to his remarks. He read his daughter's interview with great pride—though he disagreed with much of it—and also with great jealousy. When he called her, he couldn't help showing his hurt a little.

"Well, darling, you got three times as much play as I did."

"Oh, Dad. That's because of all the photographs they used. And after all, a successful woman's still news. There aren't that many yet. They wouldn't pay me a minute's notice if I weren't your daughter. They still call you 'The Money Magic Man,' you know. They know you're a genius. Beside you, I'm just a hack."

"Well, that's a lie," Steady said, "but it's nice to hear you say it."

Julia never married, perhaps because it was no problem for her to pick up men—there were plenty of handsome parasites around, Steady knew. At the same time, marriages had gone rather swiftly out of fashion, including Steady's own, and Julia was distinctly a child of her generation. She was also distinctly a child of her mother. Steady admitted to his shrink—in the days when he had time for a shrink—that he would have been just as happy, maybe even happier, if his daughter had inherited less of her mother's whacky creative genius and more of her good looks. He also had been forced to confess that it was the end of "traditional conjugal relationships," as the shrink put it, which accounted for much of Companion Pieces' success. If he lamented the demise of marriage, Steady could also profit from it with his surrogate love machines. And why did he choose to invest his business talent in this industry of all others? This was dark territory, and it took Dr. Silver two years to guide Steady to the horrifying revelation that the female Companion Pieces rather nicely resembled the daughter (and the *kind* of daughter) he really wanted to have, and the male robots resembled his own ideal physical self more than just a little. It was worse than that. The intimations of incest, at least on the level of robot companions, was nothing compared to another little story that even Dr. Silver never learned.

It was after the divorce, and yet in those more spacious times when the profit margins were not so squeezed, and before Steady found himself lending limbo money to keep his shaky conglomerate glued together. He missed his former wife terrifically and, when even Silver's ministrations were insufficient to ease his torment, he went to visit her. Marcia didn't especially mind having him around for short periods, but the clear, and to Steady, painful fact was that she was not really interested in him. He was just sort of *there*. She was far more interested in her computer art.

The irony of this was that Steady had been the one to introduce Marcia to the possibilities of the medium, and had advised her on developing programs, and on purchasing various kinds of "output." She started off with a little plotter, which could be programmed to draw all kinds of lines, such as sine waves, and intersect or overlap them. Then she graduated to a system which would "type" dots or xs in groups or lines to produce various densities and shapes of gray, and sometime later added to this a scanner which could select and reproduce on any of the output systems predetermined levels of gray in any photograph or printed image. On top of that, Marcia's computer could store for later use any image, in part or in whole, it ever created. Marcia became an addict.

The art, and not Steady, became her passion.

On his last visit, Marcia had just installed a color-video console and a color scanner.

"Someday I'm going to have color output, too," she said. "But right now the best I can do is to photograph the images I want from the tube. Of course, with the videotape, I can run through a whole series of variations and choose from those."

"So what are you working on? That's a lot of hardware for a hobby."

"Who passed a law you have to make money from these things? Besides, if you want to know, I've been selling some of my work. And I don't care about that. That's your hang up."

"You still into those op-art geometrical things?"

"No. I got tired of intersecting sine waves a long time ago. You want to know what I'm doing, really? I'd show you, but I'm not sure you can handle it. Self-portraits."

With his insistence, Marcia showed him how she worked, and unfortunately for Steady, since she was her own model, she disrobed. He suffered quietly through the whole demonstration.

"I've got three scanners here," she said, turning slowly on a dais. "Four, counting you. This one reads a special frequency of blue-green and pale orange. It's the color of your arteries and veins. See? I'm trying to pick up how much light is reflected from inside back through the skin, to see how much of the bloodlines show through. Scary, huh? This other one's an infrared sensor. Actually, it picks up heat signals, and shows them on the tube as red. Where the body temperature is hottest, the color is brightest. This last scanner picks up gray, about thirty percent, you know, just for contours. See what I get? A very abstract portrait, but it's perfectly true to life. It's just one level of me, one layer in the whole physical aura, one that's there but unseen when mixed in with all the others. You ought to see yourself at forty percent gray, Steady. You'd never be the same again!"

The encounter left him shaken. "Is she nuts or am I stupid?" he wondered. He remembered vividly the best of times they had had together and then, with Silver's help, recalled the worst times too, when their love life didn't match either of their expectations. And, newly deprived of love, Steady became a little short of obsessed with

Talk about waste-conversion processes became the sine qua non *of kaffee klatch and cocktail party.*

its dynamics, especially those most amenable to analysis, the physical ones. He borrowed techniques from the new breed of computer coaches and took high-speed films of couples in the act of love. These he fed into the computer frame-by-frame, and had them analyzed according to a whole parameter of stresses and strains, pushes and pulls, gripping and grinding, shifting and shaking. The research had two results. The first was the business that made him a millionaire: a computer-sex-counselling service in which couples with problems were filmed at high speeds and analyzed according to the results of Steady's first researches. He had referrals from many psychiatrists, including his own, and was instrumental in establishing the ethical codes and professional standards for the licensing of computer-sex-counselling practitioners of which, of course, he was the first.

The female robots had an inner glow—a gentle body heat stolen directly from the self-portrait of Steady's former wife.

The second result was the business that would make him a billionaire. It happened, in part, by accident. Steady and his researchers had created for the purpose of computer analysis what they called "the perfect loving couple." This was their idealized composite, the necessary standard of comparison used in the instruction of their clients. To move from this scientific cartoon to actual, functioning models was the next logical step.

It can't be done, everyone said. In spite of the odds, Steady began pounding his sex-counselling fortune into the research that would make "the perfect loving couple" a demonstrative, mechanical-electrical reality that could in turn, be exploited to teach sexual fulfillment to every couple on the planet. And, to the amazement of the scientific world and the newspapers and newsweeklies, Steady and his researchers succeeded. And just in time, as far as Steady was concerned, for he was almost broke.

It was Julia who first alerted him to the perils of his devotion and the investment of his fortune.

"But people aren't conjugating any more," she said. "Sex is dead."

Involved as he had been in his work, he had not realized the extent to which society had changed. "Why aren't they conjugating?" Steady demanded of his daughter. "How can this be?"

"It's too much trouble," she replied. "Who wants to be bothered?"

Bothered! The research team went back into action. Keeping his business sense about him, Steady was shrewd enough to add to his scientific staff some really ruthless market analysts. A year later, as "the perfect loving couple" sat gathering dust in Sex Consultation Laboratory forty-four, Steady met with his researchers again. After hours of discussion, the Nobel laureate Arthur J. Kroswine summed up the proceedings in a single sentence: "What we have here is widespread social demand for the ultimate easy lay."

The next morning, Steady made his decision. "The perfect loving couple" would be separated, and the male and female robots would be separately mass-produced. Steady Stedmund was going to make an assault on American loneliness from which it would never recover. He knew it was a sellout of his scientific life and principles, but was it his fault that people no longer appreciated the pains and pleasures of love?

After that, the business simply turned into refining the Companion Pieces concept. Some of the robots had radios installed: some with channels that could be tuned to any popular station, and others with a private channel tuned to Companion Pieces own broadcasting center. The Companion Pieces studio devoted itself to creating ultimately desirable dialogues, customer flattery, and specialty hours for those who wanted their robots to voice certain opinions or feelings. Where else could you receive the "Kiss and Make Up Hour", or the "Don't Feel Bad, Darling Show"?

As Play Dolls and other companies came into competition, the Companion line gradually expanded to include a wide variety of command-response programs, coin-operated models popular with resorts and hotels, and a whole field of accessories, including fashions, car rentals, vacation plans, insurance packages, and game shows. Until he discovered limbo money, out of financial necessity, Steady felt quite a bit like a business robot himself.

What Dr. Silver never learned was the Companion Pieces success secret. Why did the company stay on top of the field? The male robots held their own with the competition, but the females outsold the Play Dolls version by an astounding margin. Companion's female robots, whatever they wore, whatever the color of their eyes or hair or skin, seemed to have a very lively inner glow—as Steady knew very well—of artery and vein, a lovely shape, and gentle body heat stolen directly from the self-portrait of his former wife. He had made a harem of her, and he was glad. She was not so special any more. Anybody could have her. Psychologically speaking, he had his wife—and his daughter—right where he wanted them.

The difference between the two women was that, to all appearances, Julia remained innocent of the resemblances while her mother recognized them right away.

Marcia never forgave him, and he never saw her again. Not even by accident. They had to put her away.

Steady had just finished engineering a series of short-term limbo-money loans that had all gone badly, leaving him exhausted from juggling budgets, trying to keep up with the computer, when he was given a call from his West Coast manager, Bob Rawlings.

"Brace yourself for this one, Steady," Rawlings said. "Our main East Coast distributor wants to drop our line and pick up Play Dolls."

"Who? T. Frisch and Co.?"

"The same."

"They can't do that. What is this? Last year they signed an exclusive franchise agreement. They got about thirty percent of our whole inventory out there!"

"They're doing it. Actually, we're not even supposed to know. There was a leak."

"Jesus! What do they want?"

"Well, as I get it, they're stretched pretty thin and Play Dolls has simply offered them a bigger commission on sales, plus free franchise rights, and stock options."

"Can we stop it?" Steady asked.

"It looks bad. Everything's so secret. I've tried talking to Frisch directly, but all I can get from him is that he's not happy with us."

"Let's buy the bastard out."

"I broached that. But he's not interested in selling, at least not to us. He says we're too shaky. You won't believe the figure he set. He must have been joking."

"Let's hear it."

"$3 billion."

Steady leaned back in his chair, a little dizzy. He hated being blackmailed by a guy like Frisch. But there it was. He might have been prepared to buy Frisch for $20 million at the outside. $3 billion? That was insanity.

"Is anybody else making an offer?" Steady asked.

"I'm not sure, but I think so. I heard your daughter was after it."

"Julia? Great! Isn't she fast, though? Does Frisch know it's her?"

"I'm not sure he does, but look, Steady...uh. I don't know how to tell you this, but I think Julia was in there *even before* Play Dolls made its move. That's the way I put it together."

"That doesn't make sense," Steady said. "Unless she knew Frisch was going to double-cross us."

"You'd better talk to the horse," Rawlings said. "Meanwhile, I'll stall Frisch in court if he tries to dump us. It's about the only move I've got left."

Steady contacted Julia at once.

"Darling, what do you want with Frisch?"

"I thought you'd never ask," she said. "He's just too big to be trusted, that's all. I thought I'd try to move him out before he blackmailed us, but it looks like I started a landslide."

"Did he name a price?"

"I think I can buy him for $2 billion."

"That's outrageous. Ridiculous. Does he know you're the buyer?"

"Actually, I think I've stayed out of sight pretty well. He might be suspicious, but he can't be sure."

"What do you think, darling?"

"I think if we don't buy it and suffer for a few years, it's going to go to Play Dolls. I have an idea Frisch is their last chance. We take Frisch, we take Play Dolls too, for free. They die, we come off with scratches."

"It's a great world, eh, baby?"

"Don't worry about me, Dad."

"How do we do it? Get Frisch, I mean."

"You'll have to swing a big limbo loan for the interest. The biggest ever," Julia said. "My holding company will take your stocks, as usual. We'll have to be able to squeeze out $25 million to guarantee the bonds."

"You're not thinking of making the distributorship public and selling stock? I want it outright. Those dividends will kill us."

"It won't be so bad with Play Dolls out of the way, will it? And to take it outright, you'd have to put all of your stocks in limbo for, let's see, somewhere around thirty hours."

"$750 million, right?"

"Right."

"You love me, darling?"

"You're my dad, aren't you?"

"All right. Let's do it. You get my stocks on a delayed transaction, which gives me $750 million for thirty hours. This I loan for thirty-hours interest. The interest is passed through you to make the down payment on Frisch's distribution center."

"Some people call that 'laundering,' Dad."

"Well, that's just how we're going to wash Play Dolls out of our hair. Set it up, darling, and when you give the word, I'll put my stocks in limbo."

"It may take a day or two. Things aren't done at computer speed at that level."

"You just give me the word. Goodbye, love."

"Goodbye, Dad."

After he cleared his head, Steady called Rawlings and said simply, "Just keep Play Dolls at bay, Bob. We're going after Frisch through the back door."

"Watch yourself," Rawlings said.

"No sweat."

At the end of the next day, Julia called to let Steady know that Frisch had been set up for payoff at 11:00, day after next. Immediately, Steady "sold" his stocks to Julia, and instantaneously routed the value of the stocks through a loan channel that would place them in the hands of a wealthy New York realtor for thirty-one hours before they would be credited to Julia's corporation, and then "resold" to Steady. Just by moving money from one place to another a little more slowly than it might otherwise travel, Steady was generating enough interest to make the down payment on Frisch's distributorship without cutting into any of Companion Pieces' capital or assets. If you were rich enough, it was the ideal system for making something out of nothing. "Money is prolific," the old textbook adage went. But nobody, until Steady, had realized just *how* prolific it could be. For the first time in years, Steady took a break from his corporation work.

"I'm taking a thirty hour vacation," he told Rosa. "Don't bother calling, because I'm going to get lost."

"You need the rest," Rosa said.

"And for once, I think I can afford it," Steady said.

He went straight to an exclusive, private restaurant at the top of one of the city's most famous buildings. Shrimp. Peking Duck. Orange Sauce. Wild Rice. Fried apples in Himalayan honey. An ancient bottle of sparkling burgundy. And then to bed for the first good sleep in years.

Thirty hours later, he sat by the phone in his office waiting for Julia to call and say, as usual, "It's done." How he loved to hear his loving daughter's voice. And this time her simple announcement would mean so much more than it ever had before: Companion Pieces would be free of competitors and free from any interference in distributing its goods. Best of all, Steady could relax from his hectic life with LIMAR and finally move on to something new, perhaps even politics. He had clout, and Senator Wilton was losing popularity with several of the party leaders.

The phone rang right on time.

"Hello, darling."

"Hello, Dad," Julia said. "You're fired."

Steady exploded in laughter.

"It's not a joke, Dad. It's a coup."

Steady couldn't stop laughing. *Yes, of course,* he thought. He certainly had trained her well. He laughed and laughed.

"Our check for $750 million, less agency fees, will arrive in about forty-eight hours," Julia said. "You have to admit, that's a nice nest egg for your retirement. If you want to invest it, you know, we'll be glad to help you out."

"I'm sure I couldn't find a more capable firm," Steady managed. "How does it feel to be president of Companion Pieces, anyway?"

"Just great," Julia said. "It's what I've always wanted."

Steady let the phone fall to the floor. ▼

Cryptic Computer
WORLD WAR II'S LEGACY TO DIGITAL TECHNIQUES

by Frederick W. Chesson

Generally, anything that can be digitized can be subject to computer operations. As discussed in the February issue of *ROM,* the degree of machine sophistication need not be extensive. Simple examine-compare-replace techniques are entirely suitable for a majority of applications. Indeed, the earliest electro-mechanical devices for cryptanalysis were IBM punch-card readers and tabulators with varying degrees of modifications. The U.S. Army and Navy began to make use of such equipment in the mid 1930s, and by the end of World War II, highly-specialized apparatus had been developed, some employing considerable numbers of vacuum tubes. The largest "precomputers" were devoted to the field of Ordnance, where ballistic calculations were climbing, as if on their own trajectory, away from the limitations of the existing electro-mechanical analog machines.

The first of these was ENIAC (Electronic Numerical Integrator And Computer) from the University of Pennsylvania's Moore School of Engineering. ENIAC was fast, compared to relay-operated devices, but its 18,000 tubes operated upon a mere twenty numbers, and lacked the essential capacity of internal programming. ENIAC was followed by such machines as EDVAC, BINAC, and even MANIAC. Although each of these progressively improved upon its predecessors, all were postwar developments.

Cloaked, until quite recently, by wartime-engendered security wraps was a British creation whose scope and physical size well deserved its name: COLOSSUS. These COLOSSI (some ten were ultimately constructed) were designed for one specific goal: cryptanalysis. The target was the German cryptosystems—specifically the "Enigma" cipher machine, which was used in various versions by the Wehrmacht, Luftwaffe, and Kriegsmarine (mainly by the submarine wolfpacks).

The Enigma machine utilized a set of six rotors, whose step-wise rotation produced a new internal cipher with each keypress of its typewriter-like keyboard. Not only could the individual rotors, each containing its own scrambled wiring, be set, but the connections *between* the rotors could be varied easily by means of plugs and switches. With frequent key changes carried out, the Germans considered the Enigma reasonably secure for the highest level of radio transmissions. The capture of individual machines was not deemed fatal; in fact, it was considered inevitable in the fortunes of modern mobile warfare and espionage. The complexity of the cipher and the ease of making millions of key changes was thought to negate the eventual solution of any individual message or message group.

It was to defeat this assumption, with the greatest of secrecy of course, that British Intelligence was directed. So it was that in a suburb of Bletchley, about forty miles north of London, an assemblage of diversely talented individuals was gathered. In and around the grounds of Bletchley Park (an outstandingly ugly Victorian mansion), technical developments and research were carried out amid stately trees and intruding quonset-like huts. The various personnel concerned with the actual construction of the calculating apparatus would journey daily to the British Post Office and Telephone's engineering department's workshops in the gloomy North London environ of Dollis Hill, where German bombs fell uncomfortably close.

Following the successful development of several preliminary models, one of which was called the "Heath Robinson," research and construction on the first COLOSSUS began in early 1943. When it became operational in December, 1943, it was far ahead of anything in the electronic or electro-mechanical calculating field anywhere in the world, including (as far as is known at present) the United States. Its features are quite worthy of listing here:

- Fifteen-hundred vacuum tubes (mainly type EF 36 pentodes used for flip-flops and logic gates)
- Paper tape reading at five-thousand-characters per second
- Photoelectric tape readers
- Bi-stable (flip-flop) vacuum-tube circuits
- Variable electronic storage registers
- Conditional (branching) logic
- Preset or conditionally-set logic functions
- Fully automatic operation

The effect of the COLOSSI and their associated calculating apparatus upon the course of the war has been discussed in several books. So fast were Enigma-enciphered messages decrypted, that Churchill often had Hitler's orders at hand before the German generals to whom the radiograms were intended. The greatest of care was necessary that the British and American commanders who could best benefit by the intelligence recovered would not know too much to give away such a priceless treasure. One reason for the initial success of Hitler's last great offensive in the Ardennes in December, 1944 was that the most vital orders were hand-carried, and lower-level directions were sent over reasonably secure telephone lines. Ironically, the very lack of "Ultra" reports, as the Enigma interceptions were called, tended to downgrade the importance given to visual sighting of gathering Panzer divisions at the Belgian border. Had Hitler used conventional radio links, the famed "Bulge" might have been contained by well-planned counter-attacks before it became more than a tentative convexity in the American lines.

Both ENIAC and COLOSSUS were program-controlled, electronic digital computers. The former was directed towards numerical calculations largely involving differential equations; the latter was oriented "Towards Boolean calculations of a particular type" (cryptanalytic operations). The American machine was the immediate parent of EDVAC, which incorporated a practical, stored program.

With the advent of the Cold War, which effectively sealed all the achievements of the Bletchley Park team for decades, American efforts to develop computers for cryptographic purposes were to have profoundly stimulating effects upon the data-processing industry. These are just coming into public scrutiny now, and will be examined in a future article.

RUNNING DOWN THE ALPHABET

In spite of the implication, running down the alphabet is a form of simple cryptanalysis. It may be implemented easily on a micro- or minicomputer.

In the so-called "Caesar Cipher," which was probably known before the advent of Julius on the Roman scene, each letter of the plain message is shifted a fixed number of space along the alphabet. Thus, if the key is five letters, *GAUL* becomes *LFZQ*. With a key of twenty letters, *ROMA* becomes *MJHV*. Since literacy was something of a black art to all but the educated elite, such a simple device probably served to secure the despatches from the eyes of the preliterate Alpine tribesmen.

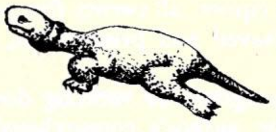
To examine the programming steps necessary to accomplish this process, an actual mini-sample of running down the alphabet is shown below.

```
Cipher: Y N U L P K C N WL D U
Step  1. Z O V M Q L D O X M E V
Step  2. A P WN R M E P Y N F W
Step  3. B Q X O S N F Q Z O G X
Step  4. C R Y P T O G R A P H Y (Probable Solution)
Step  5. D S Z Q U P H S B Q I Z
Step  6. E T A R V Q I T C R J A
..............................
Step 25. X M T K O J B M V K C T
Step 26. Y N U L P K C N WL D U (Original Cipher)
```

The sample above indicates the cyclic process involved. In order to advance the value of any given alphabetic character by one space, it is only necessary that the alphabet or character set be accessible for each character word concerned. For the normal alphabet, it can be represented as ALPHA (1), ALPHA (2), ALPHA (3), etc.

In this example, each plaintext letter has been moved forward twenty-two steps. Thus, it takes 26 − 22 = 4 steps to restore the original meaning. The program would therefore determine each letter's position in the alphabetic array by means of a loop test, and the printing out or displaying of the next letter in the array.

For purposes of encoding, the Nth letter (where N = 1 to 25) is displayed; for decoding, the 26 − N letter is recorded.



Moving?

Don't leave your

ROM behind!

Please print your name and *new* address below and mail to ROM Publications Corp., Route 97, Hampton, CT 06247

Name ____________________

Street Address ____________________

City ____________________

State ____________ Zip ____________

(Please allow 6-8 weeks for processing.)

CHANGE OF ADDRESS
Please attach your present mailing label here

Solution to last month's PROMpuzzle

R	E	A	D	■	■	I	M	A	G	E	■	L	A	S	E	R
A	R	N	O	■	■	N	A	B	O	B	■	I	R	A	T	E
L	O	G	E	S	■	P	R	E	■	C	R	O	M	■	U	S
U	S	E	■	P	O	U	T	■	E	D	E	N	■	D	D	E
■	■	L	E	A	S	T	■	D	A	I	S	■	D	I	E	T
■	■	■	A	R	E	■	M	I	S	C	■	T	O	E	■	■
W	H	I	S	K	■	P	A	G	E	■	D	E	B	U	G	S
A	I	N	T	■	S	A	R	I	■	H	I	R	E	■	A	T
R	N	S	■	B	I	T	S	T	R	E	A	M	■	S	T	A
D	I	■	S	E	N	S	■	I	S	I	S	■	P	O	E	T
S	L	O	P	E	S	■	C	Z	A	R	■	S	E	N	S	E
■	■	P	O	T	■	H	O	E	S	■	T	O	N	■	■	■
W	A	T	T	■	P	A	R	S	■	H	I	L	T	S	■	■
E	L	S	■	B	A	N	D	■	L	O	C	I	■	I	C	F
B	O	■	C	O	N	G	■	F	A	R	■	D	O	N	O	R
E	N	T	R	Y	■	U	N	I	T	S	■	■	A	C	R	E
R	E	S	T	S	■	P	A	R	E	E	■	■	R	E	E	D

When examining the unknown cipher, all twenty-five possible variations should be displayed and printed out, or stored for future evaluation.

The following FORTRAN program for running down the alphabet is taken from the author's comprehensive Cryptanalysis Program which appeared in the January 1973 issue of *Datamation.* The alphabet was stored in the program body as a data statement, but it could have been read in just as easily from the keyboard or the punch card, as was the cryptogram. For microcomputer applications, it is possible that the ASCII letter set in the CRT display's character generator could be utilized to furnish an internal alphabet and thus eliminate the need for a separate reading in.

```
      INTEGER ALPHA(27), W
      DIMENSION KRYPT(100), KTA(100)
      DATA ALPHA/ 'A','B','C','D','E','F','G','H','I','J','K','L','M',
     1 'N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/
      ALPHA(27) = ALPHA(1)
      WRITE (W,1)
1     FORMAT (1H1, //, 45X, 'CAESAR RUNDOWN', 77)
      WRITE (W,2) (KRYPT(J), J=1, 100)
2     FORMAT (5X, 100A1)
      DO 8 J=1, 100
3     KTA(J) = KRYPT(J)
      DO 7 I=1, 25
      DO 6 J=1, 100
      DO 5 K=1, 26
      IF (KTA(J) - ALPHA(K)) 5,4,5
4     KTA(J) = ALPHA(K+1)
      GO TO 6
5     CONTINUE
6     CONTINUE
      WRITE (W,2) KTA
7     CONTINUE
      STOP
      END
```

Figure 1 shows an actual decipherment employing this program. While simplification is possible, especially in conversion to BASIC or some other language, the essentials of the nested-loop operations remain true.

Here's more for the dedicated cryptanalyst to ponder while his/her program is running down ciphers.

1. "Codes on Cards"

QSNSGYKMSEW YJ DLUKWOEOGUWID
MODAIEST IE WAS VEIWSQ TWOWST
QOWST JLYM WAS SOLGU WAILWIST
OEQ WAS VTS YJ I.B.M. WOBVGOWYLT
BU WAS V.T. EONU.

Hint: Some letters may stand for themselves.

2. "Computer Ancestors"

HANDY CODES BADDC DFGBH CIKNK
LOHBF ICKBO ZCUCD WADLC YWDCX
NCIHS JANIG KOIYA GBCDK GOGFK
GFHOE AUCDO GFAIK? SCKFI YCCY!

Hint: Undivided words—five-letter groups.

For cryptanalysts in need of a helping hand, solutions to these cryptograms will be found on page 73. ▼

Figure 1
CAESAR RUNDOWN

```
NYHNBFYACIHGIPYMUAUCHMNWULNBUACHCUHMUNXUQH Crytpogram (Ciphertext)
OZIOCGZBDJIHJQZNVBVDINOXVMOCVBDIDVINVOYVRI
PAJPDHACEKJIKRAOWCWEJOPYWNPDWCEJEWJOWPZWSJ
QBKQEIBDFLKJLSBPXDXFKPQZXOQEXDFKFXKPXQAXTK
RCLRFJCEGMLKMTCQYEYGLQRAYPRFYEGLGYLQYRBYUL
SDMSGKDFHNMLNUDRZFZHMRSBZQSGZFHMHZMRZSCZVM
TENTHLEGIONMOVESAGAINSTCARTHAGINIANSATDAWN Solution (Plaintext)
UFOUIMFHJPONPWFTBHBJOTUDBSUIBHJCJBOTBUEBXO
VGPVJNGIKQPOQXGUCICKPUVECTVJCIKPKCPUCVFCYP
WHQWKOHJLRQPRYHVDJDLQVWFDUWKDJLQLDQVDWGDZQ
XIRXLPIKMSRQSZIWEKEMRWXGEVXLEKMRMERWEXHEAR
YJSYMQJLNTSRTAJXFLFNSXYHFWYMFLNSNFSXFYIFBS
ZKTZNRKMOUTSUBKYGMGOTYZIGXZNGMOTOGTYGZJGCT
ALUAOSLNPVUTVCLZHNHPUZAJHYAOHNPUPHUZHAKHDU
BMVBPTMOQWVUWDMAIOIQVABKIZBPIOQVQIVAIBLIEV
CNWCQUNPRXWVXENBJPJRWBCLJACQJPRWRJWBJCMJFW
DOXDRVOQSYXWYFOCKQKSXCDMKBDRKQSXSKXCKDNKGX
EPYESWPRTZYXZGPDLRLTYDENLCESLRTYTLYDLEOLHY
FQZFTXQSUAZYAHQEMSMUZEFOMDFTMSUZUMZEMFPMIZ
GRAGUYRTVBAZBIRFNTNVAFGPNEGUNTVAVNAFNGQNJA
HSBHVZSUWCBACJSGOUOWBGHQOFHVOUWBWOBGOHROKB
ITCIWATVXDCBDKTHPVPXCHIRPGIWPVXCXPCHPISPLC
JUDJXBUWYEDCELUIQWQYDIJSQHJXQWYDYQDIQJTQMD
KVEKYCVXZFEDFMVJRXRZEJKTRIKYRXZEZREJRKURNE
LWFLZDWYAGFEGNWKSYSAFKLUSJLZSYAFASFKSLVSOF
MXGMAEXZBHGFHOXLTZTBGLMVTKMATZBGBTGLTMWTPG
```

Get updated . . . keep updated with

BYTE

the leading magazine in the personal computer field

BYTE
BYTE
BYTE
the small systems journal

The personal computer age is here.

Join Byte's 110,000 subscribers and catch up on the latest developments in the fast-growing field of microprocessors. Read BYTE, The Small Systems Journal that tells you everything you want to know about personal computers, including how to construct and program your own computer (over 30,000 BYTE readers have already built, or bought, their own systems and half of these have 8K bytes or more).

You'll find our tutorials on hardware and software invaluable reading, also our reports on home applications and evaluative reviews based on experiences with home computer products.

Home computers . . . practical, affordable.

Large scale integration has slashed prices of central processors and other computer components. This has encouraged the development of new, low-cost peripherals resulting in more hardware and software — more applications than you could imagine, more opportunities for you. BYTE brings it all to you. Every issue is packed with stimulating and timely articles by professionals, computer scientists and serious amateurs.

BYTE editorials explore the fun of using and applying computers toward personally interesting problems such as electronic music, video games and control of systems for alarms to private information systems.

Subscribe now to BYTE . . . The Small Systems Journal

Read your first copy of BYTE, if it's everything you expected, honor our invoice. If it isn't, just write "CANCEL" across the invoice and mail it back. You won't be billed and the first issue is yours.

Allow 6 to 8 weeks for Processing.

© Byte Publications, Inc. 1977

BYTE Subscription Dept. 85 • P.O. Box 361 • Arlington, Mass. 02174

BYTE

PLEASE ENTER MY SUBSCRIPTION FOR:

☐ One year $12 (12 issues) ☐ Two years $22 ☐ Three years $32

☐ Check enclosed (entitles you to bonus of one extra issue)

☐ Bill me ☐ Bill BankAmericard/Visa ☐ Bill Master Charge

Card Number: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date: ____

Signature: ____ Name *(please print)* ____

Address: ____

City: ____ State/Country: ____ Code: ____

FOREIGN RATES FOR ONE YEAR: *(Please remit in U.S. Funds)*

☐ Canada or Mexico $17.50 ☐ Europe $25 (Air delivered)

☐ All other countries except above: $25 *(Surface delivery)*

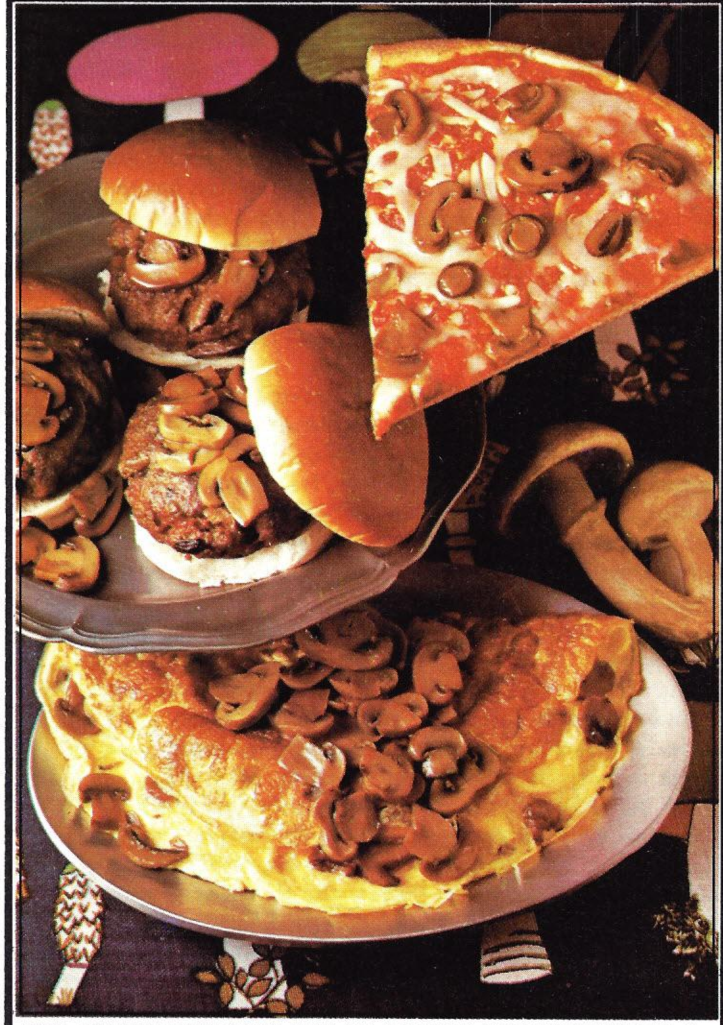
Air delivery available on request

COLLIER IS THE BEST ENGRAVER, PRINTER IN THE COUNTRY.

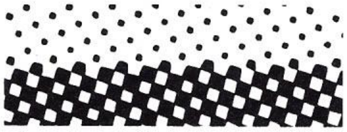
Because. A revolution in engraving has happened. Collier split the dot. And because of the new Laser Beam, Collier is now light years ahead of the rest. The laser does it. Here's how it works. Technically, the laser beam which is used to control film exposure (thus "Program" the engraving) is split into six sectional beams. Each of these is digitally modulated by computer to transfer half-dots of picture information per scanner revolution. Since two picture-information "bits" are available per dot in circumferential direction, it's possible to expose a smaller area in the second "orbit"...or even completely omit it (thus producing an actual half-dot or even an elliptical mini-dot). If that seems to all sound a lot like gobbledygook, don't worry about it. The important thing is, it's here and it does work and it gives your image resolution that you never could get before. We'll be happy to demonstrate it for you. Even happier to produce your next set of four-color letterpress, offset and gravure engravings. Call Collier Graphic Service Company Inc., 240 West 40th Street, New York, New York 10018/(212) 840-0440.

ATLANTA	BOSTON	DETROIT	HARTFORD	NEW YORK	PHILADELPHIA
(404) 892-2383	(617) 965-5660	(313) 259-2111	(203) 367-0706	(212) 840-0440	(215) 988-0110

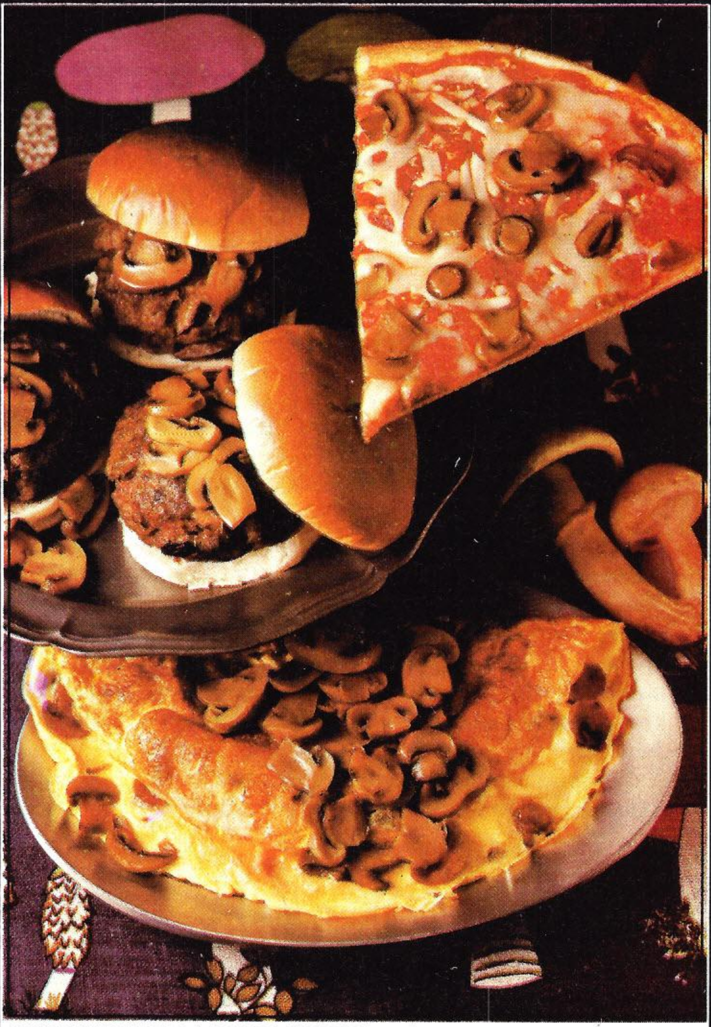
OR CALL TOLL FREE (800) 221-2585/(800) 221-2586



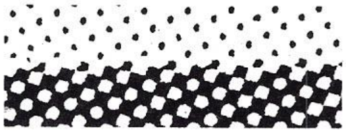
THE ABOVE ENGRAVING WAS MADE WITH THE CONVENTIONAL PLATE MAKING PROCEDURES, AS YOU CAN SEE, IT LACKS COLOR FIDELITY AND SHARPNESS, IF YOU COMPARE IT WITH...



THE CONVENTIONAL ENGRAVING DOT.



THE COLLIER'S LASER BEAM ENGRAVING, AS YOU CAN SEE FROM THE ABOVE, HAS BRILLIANCE, SHARPNESS AND COLOR FIDELITY THAT ONLY COLLIER'S DOT CAN PROVIDE.



THE COLLIER LASER DOT.

THE COLLIER ADVANTAGE. COLLIER GRAPHIC SERVICES COMPANY, INC., 240 WEST 40TH STREET, NEW YORK, NEW YORK 10018

futuROMa

THE CAT COMPUTER: SCRATCH, PURR, FETCH DATA

by Bill Etra

What do we really want from robots? Asimov hypothesizes a Frankenstein fantasy that does not allow robots to become popular on earth, but only in the colonies on the stars, where people really need them. Is this their only place in the web of technology? How close are we, really, to having robots at our personal disposal? And a most interesting though rarely asked question is, what do we want robots to look like?

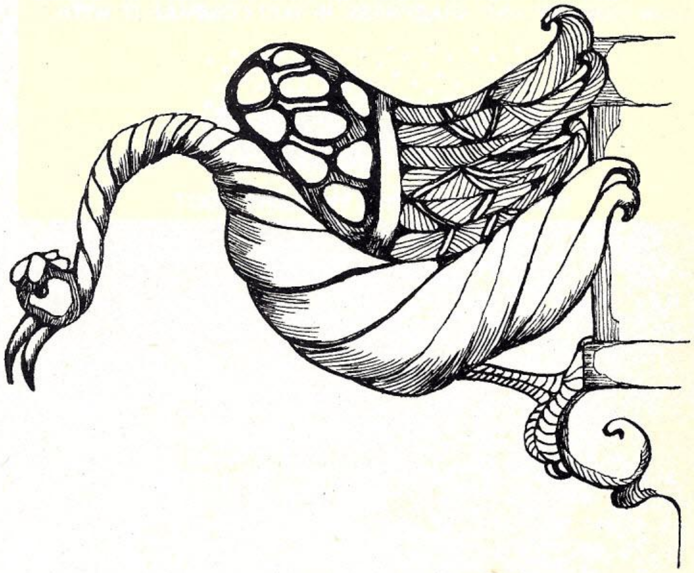
In point of fact, there are a number of companies around the world that have been manufacturing various types of robots for several years. They range from remote-controlled intelligent mechanical brooms that run around the floor sweeping and polishing to full servants that look like something out of a bad version of Flash Gordon and can do anything up to and including mixing martinis. The catch is, they are very task specific—like the automated pinsetter at the bowling alley, which is essentially a robotic device—and not at all the general-purpose robot of which we dream.

In our dreams and in science fiction, we think of creations more like Asimov's robots in *Caves of Steel* and *The Naked Sun.* Here the robot is a human lookalike which can do everything, including eating or at least depositing its food neatly in a plastic Ziploc bag which is removable from its chest cavity. Replete with plastic skin, synthetic eyeballs, and so on, it is the logical extension of the Six Million Dollar Man. The two most famous robots of the moment, See-Threepio and Artoo-Detoo of *Star Wars* fame, fall into the same group, though here we have an interesting anomaly. We have one robot built like a human that has to talk to another robot built like a garbage can with rollers. Yet Artoo-Detoo, which resembles nothing so much as my industrial vacuum cleaner, in my eyes is by far the more personable of the two.

To a great extent in their brief history, it's been thought that robots would become copies of humans and/or animals. Yet, with the possible exception of the Disneyland dummies and other animated displays, most of the functional robots haven't evolved a human form. Take that example of the mechanical pinsetter, for instance. It doesn't run around your bowling alley looking like the old pin boy, setting your pins up with mechanical arms. True, occasionally industrial robots do have human-looking hands. But most of that is for show.

So what about more of the science-fiction Six-Million-Dollar-Man, Six-Million-Dollar-Woman, Two-Million-Dollar-Dog type robot? Well, the major problem with a humanlike robot is that the human body is a very general-purpose device. It needs very complex programming in order to perform any special function. Digital programming, as we know it today, simply doesn't lend itself to this complexity. To program a humanoid body with all the gestures and movements necessary to even walk through a room avoiding people is, despite the thoughts of modern movies like *Westworld,* a staggeringly difficult task.

A way around the programming problem has been suggested by various science-fiction writers. The Australian science-fiction writer Cordwainer Smith suggests that animal brains be used to drive the robots. While we have very little information on the complex programming of biological forms, we do know a lot about animal training, particularly Pavlovian conditioning.

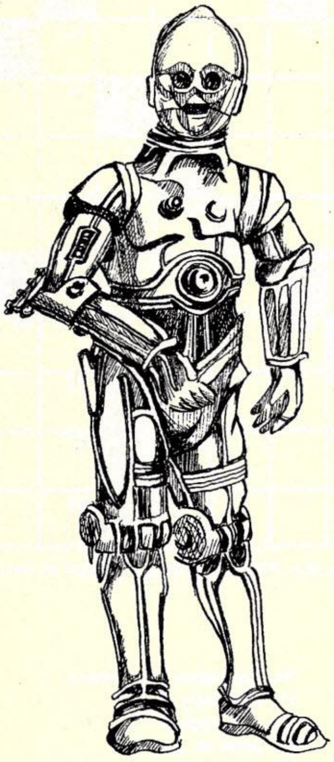


Consider, for instance, my black cat Orpheus. Orpheus has spent the last ten years ruling the lower biosphere of my home, and—not to be overly morbid—he may live another ten. But at some point he'll disappear. All that love and

Illustrated by Linda Smythe

energy and affection and time spent studying his behavior while I stared at the wall trying to write articles for *ROM* will have gone to waste. But what if I could encode Orpheus's personality on my computer, either through some direct electronic means or by preserving his brain?

If his brain could somehow be biologically or electronically connected to a computer, we could continue our long-standing, ongoing relationship. Certainly that is one of the directions in which technology is heading. Consider the future when we get a field-effect device that makes it unnecessary to connect all the little individual nerves into the machine. The brain can then sit in a field that monitors all the nerve endings. Well, in that case, Orpheus would probably make a wonderful computer. All I'd have to do would be to simulate a scratch behind the ear or his favorite kidney-flavored catfood, and Orpheus would probably be very glad to do anything I asked him to. Meanwhile he, with his mind inside the machine, would rest in a perfect cat world, on a large soft pillow in front of the fire, with an occasional mouse running by and getting caught in an imaginary claw, with me scratching his back or his stomach.... Possibly I'd withhold some pleasure while he went to fetch something from yesterday's data bank, tomorrow's schedule, or Saturday's billings. The brain in the computer—cat brain, mouse brain, squirrel brain, dog brain, your favorite pet preserved forever in a useful and conditionable form—may be the only way to get around the present programming problem—the only way to arrive at, for instance, the super-automated house of the thirties, that science fiction of home computing which most popular magazines have seen the home-computer revolution as heralding.



I am reminded of nothing so much as of a Ma and Pa Kettle movie. The last thing on the Late Late Late Late Show when I lived in New York and was up all night working, and using the television signal to synch my equipment. Ma and Pa Kettle inherit the house of the future. There's this nightmare of a couch that turns into a bed that turns into a bathtub and drowns people, with television screens falling out of the ceiling and automated drinks coming up on strange little mechanical clutchyhands.

This art deco, art nouveau dream of the futuristic, Artoo-Detoo-peopled house is really *not* something that's about to happen from the home-computer revolution, no matter how many interfaced devices may turn your stereo on and off with vocal commands. The amount of programming required to make something that would replace my cat, let alone somebody who can clean up the house, is beyond us at the moment.

But it does exist in preprogrammed form. You can train a dog. A good many of the dog's limitations are in terms of its physical build. That being a "mechanical" limitation, modifying the build might be easier than solving the software problem.

Beyond all that, what about me? If I'm going to preserve my pet cat Orpheus as the computer, when I pass on I might want to move into his space. Certainly I could simulate a very nice world. I could program my own world into my own machine and live in machine space. Why would I need limbs and the other accoutrements? If I were to stimulate my senses correctly, I should be able to create any world I want. My friends could join me. We could have limited-access spaces for privacy and shared memory for group experiences. We could travel our own universes, creating them for each other.

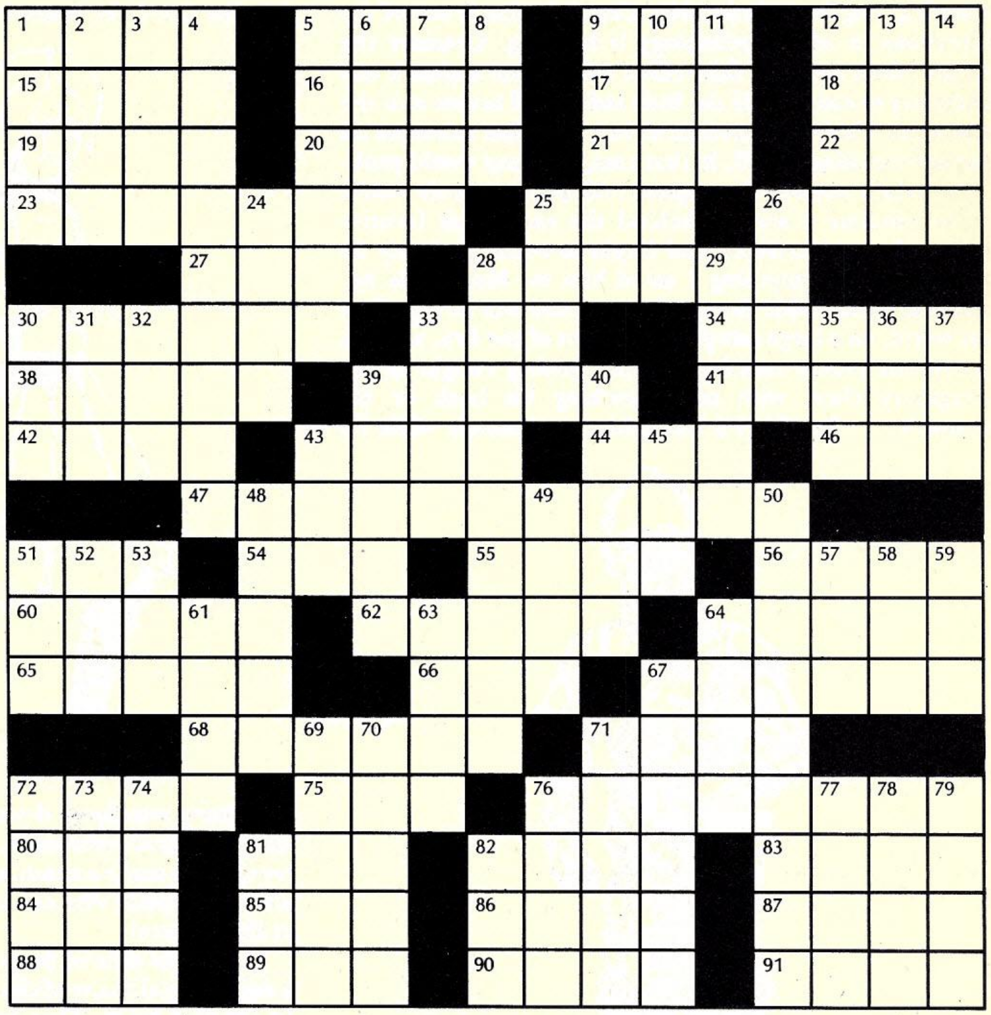
A horror, you say. Little brains inside of a great big gray or black box. But how would we know? How would we know we hadn't yet become God? Certainly in our own universe we would have become so, with our ability to create our own planets, our own environments, our own interactions. Of course there'd always be the danger of dissident factions within your computer eventually inventing that great world-ending device—the core dump. ▼

PROMpuzzle

by Daniel Alber

ACROSS

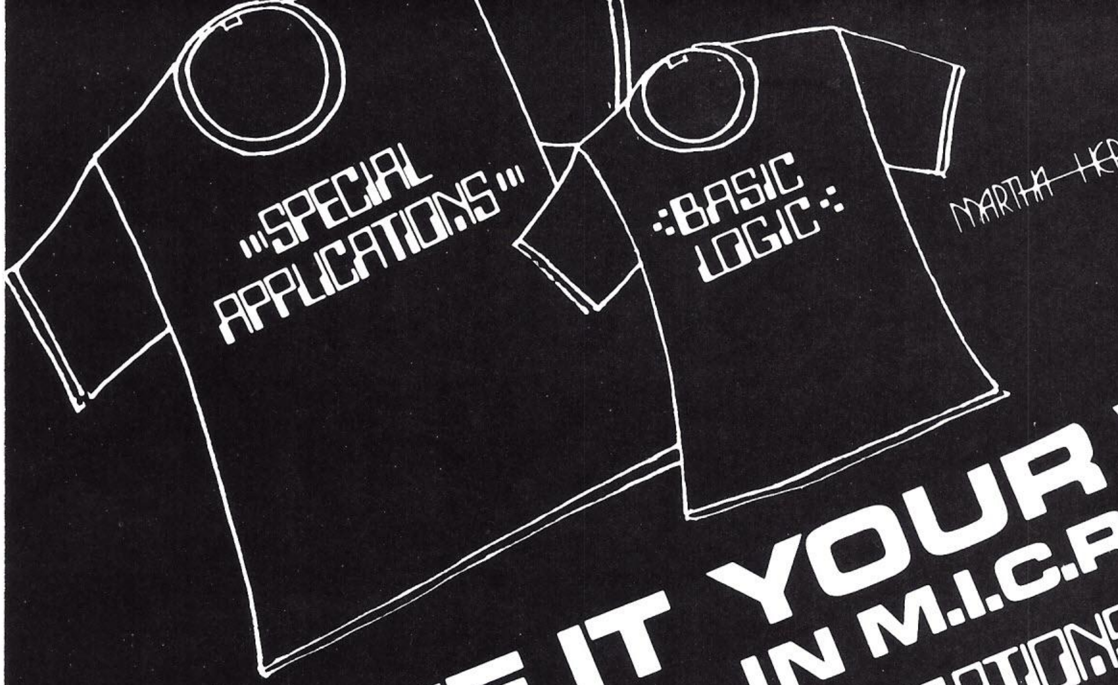
1. Always
5. Binary indicator
9. Field effect transistor (abbr.)
12. Common suffixes
15. ____ wrappable module boards
16. Actual
17. ____ mode (2 wds.)
18. ____, chip
19. Greek letters
20. American Beauty
21. ____ cage
22. Frost
23. ____ gate transistor
25. Mrs. Nixon
26. ____ converter (3 wds.)
27. Romanian coins (var.)
28. Computer designs
30. What some modules do
33. Problem's answer (abbr.)
34. Helpers
38. Powered a boat
39. Chain or column binary
41. Allowed values of a function
42. Joust
43. Indian dialect
44. Electrocardiogram (abbr.)
46. Drunkard
47. Logical unit which adds binary words (2 wds.)
51. Stored program element (abbr.)
54. Compass reading
55. Woe is me!
56. ____ constant
60. Vital organ
62. The devil
64. Strange
65. ____ burst
66. ____ and ink
67. To keep data
68. Path around a circuit element
71. Obtains (dial.)
72. "Rock of ____"
75. Lyrical poem
76. Semiconductor impurity (2 wds.)
80. Landed
81. Before (prefix)
82. Husband (Fr.)
83. Je vous ____
84. Gone by
85. Central transfer point (abbr.)
86. Actor Guinness
87. Volume
88. Ronald, for short
89. Controversial plane
90. Volcano
91. ____ bit



The solution to this PROMpuzzle will appear in next month's *ROM*.

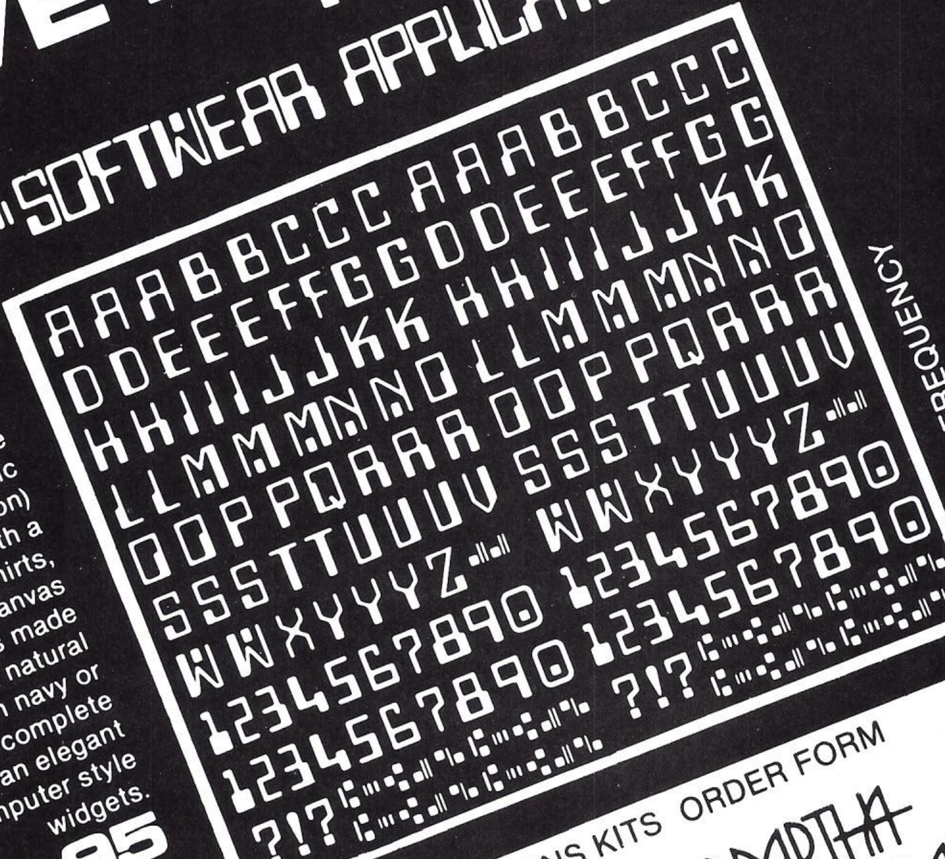
DOWN

1. Pitcher
2. Quick! (Fr.)
3. Epochs
4. Some transformers
5. Resonance or radiation
6. Presses
7. Sudden blast
8. Born
9. 1 coulumb = 1 volt
10. Select few
11. Bill
12. ____ commands
13. Puerto ____
14. Pit
24. Want
25. One electrode of a battery
26. Continent
28. Adjusts signals to a required standard
29. ____ scale integration
30. Two (Sp.)
31. Knock
32. A Gershwin
33. Scotch and ____
35. Decimal number system (abbr.)
36. Part of the psyche
37. Place a binary cell in tne 1 state
39. Hollers
40. Type of auto
43. Ornamental vase
45. Clock drivers (abbr.)
48. First instruction in a subroutine
49. Actor Ladd
50. Resistance components
51. That lady
52. ____ capita
53. Lend an ____
57. Arrival (abbr.)
58. Fib
59. "____ Miserables"
61. Steals
63. Part of a church
64. Feminine suffix
67. Computer gel
69. Input-output ____
70. Skilled
71. Bridge expert
72. With wings
73. Garbage-in, Garbage-out
74. British school
76. Drop dead ____
77. Melee
78. Bullets, for short
79. Chirp
81. Photoconductors (abbr.)
82. Miss West


"SPECIAL APPLICATIONS"
BASIC LOGIC
MARTHA HERMAN
114 W 17 STREET
NEW YORK 10011

HAVE IT YOUR WAY IN M.I.C.R.

"SOFTWEAR APPLICATIONS KITS"


LETTER FREQUENCY

Let the world know you're TEMPORARILY CRASHED or HARDWIRED or a COMPUTER SIMULATION with easy to use heat transfer MICR (Magnetic Ink Character Recognition) typeface. Apply with a household iron to shirts, jeans, tote bags, canvas luggage, or any items made of at least 50% natural fiber. Kits come in navy or white letters with complete instructions and an elegant array of computer style widgets.

$3.95
2/$7.
Add 50¢ per kit for shipping.

SOFTWEAR APPLICATIONS KITS ORDER FORM

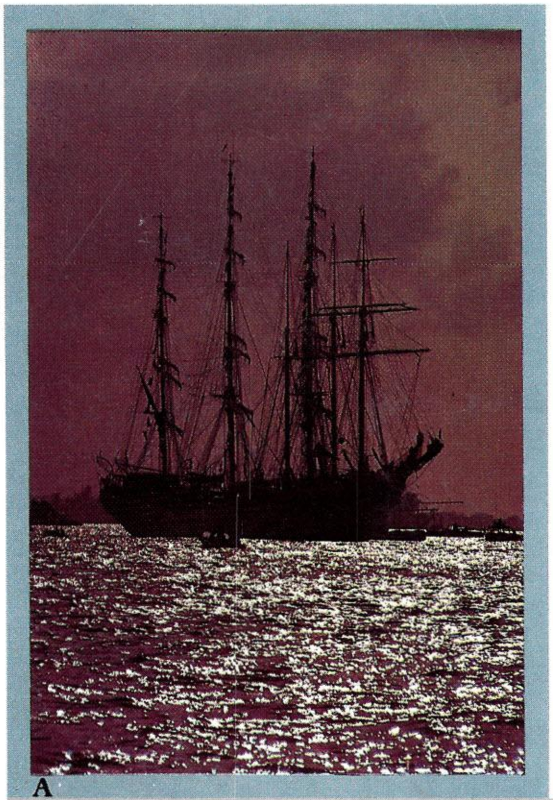
NAME
ADDRESS
CITY
ZIP
APT
STATE

mail to:
MARTHA HERMAN
114 W.17 ST NEW YORK 10011

COLOR	QUAN	$
NAVY		
WHITE		
Add 50¢ per kit for shipping		
ENCLOSED TOTAL		

A magnificent first edition . . . pictorial memories of a moment America will never forget . . . or see again

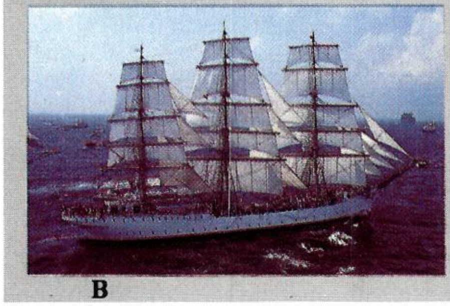
THE SAILING SHIPS


A

Professional quality lithographs of the majestic Bicentennial Ships, now available to the public in an exclusive, limited edition from America's foremost engraver-printer, COLLIER GRAPHICS.

handsomely matted
in silvery metal frames; $25 each

unframed, $10 each


B

C

D

THIS IS A LIMITED EDITION . . . ORDER TODAY WHILE THEY LAST

Never before offered to the public, these magnificent 12x19 full color lithographs of the tall-masted SAILING SHIPS were originally photographed for professional use only!

Now, you can own and enjoy their historic beauty, their incredible clarity, color and reproduction, which gives them almost a three-dimensional quality. And they are only available from COLLIER GRAPHICS — who provide the superb color graphics for America's leading advertising agencies and publishers.

Perfect for your home, boat or office. A lasting gift. Order a set for yourself and one for your children . . . to keep always.

COLLIER GRAPHICS INC., 240 West 40th Street, New York, New York 10018

Please rush the following SAILING SHIPS, securely packaged and postage prepaid, with the understanding that my purchase is UNCONDITIONALLY GUARANTEED by you if the order is returned within 15 days of delivery:

Send me the following print(s). Quantity ________ Price ________ Total ________

A. Christian Radich ________

B. Danmark ________

C. Kruzenshtern ________

D. Eagle ________

Please add $2.50 for shipping and handling for framed print(s) or $1.00 for unframed print(s).
(N.Y. State residents add applicable tax, NYC residents add 8%. Allow 6 weeks for delivery.)

Name ________

Address ________

City ________ State ________ Zip ________

Enclosed, check or money order for $________
Or charge my credit card ________ Master Charge ________ BankAmericard

Credit Card # ________ Inter Bank # ________ Expiration Date ________

Signature X ________